ロボットによる運動制御の自律学習

修士学位論文

ロボットによる運動制御の自律学習

東北大学大学院 情報科学研究科

システム情報科学専攻 篠原研究室

博士課程前期二年

大崎 嗣豊

2008年2月20日

目次

第1章

序論

] 近年の科学技術の進展により，さまざまなタイプのロボットの開発が進められている．特に，自ら状況判断をしながら行動を決定する自律型ロボットは，人間に代わって仕事をする存在として，産業分野のみならず，危険作業，災害救助，医療，家事など様々な場面での活躍が期待されている．我々は 2003 年から RoboCup サッカーの四足ロボットリーグにチーム JollyPochie[1] として参加している．RoboCup[2] は人工知能とロボティクスの融合，発展を目指し発足された国際プロジェクトである．

RoboCup サッカーには，シミュレーションリーグ [3]，小型ロボットリーグ [4]，中型ロボットリーグ [5]，四足ロボットリーグ [6]，ヒューマノイドロボットリーグ [7] の５つのリーグが存在する．この中でも四足ロボットリーグは SONY が提供する AIBO を (ERS-210, ERS-7) を用いなければならないため，全てのチームのロボットが同じ性能になるという大きな特徴を持っている．そのため各チームのプログラムの優劣のみが勝敗を左右する．図 1.1 および表 1.1 に四足ロボットリーグで使用されている AIBO の基本的なスペックを示す．

[8] では AIBO を用いて１次元上で動くボールをキャッチする技術の自律学習を行ったものを，本論文ではそれを拡張して２次元で自由に動くボールを止める自律学習を行う．１次元ではまっすぐ向かってくるボールに対していつ捕捉動作を行えばよいかという捕捉動作を開始するタイミングを学習していたが，２次元に拡張するためにはボールを止めるためにモーションを開始するタイミングを学習すると同時にボールを止めることのできる位置への移動を考えなければならない．

二次元で動くボールに対して技術を獲得する学習はシミュレーションや中型ロボット

表 1.1: AIBO(ERS-7) の基本性能

CPU	64 bit RICS プロセッサー (576MHz)
主記憶	64 MB
プログラム供給媒体	AIBO 専用メモリースティック
可動部（合計 20 自由度）	頭 … 3 自由度
	口 … 1 自由度
	脚部 … 3 自由度× 4 脚
	耳 … 1 自由度× 2
	尻尾 … 2 自由度
画像入力	35 万画素 CMOS イメージセンサー
画像解像度	208 × 160 ピクセル
画角	水平 56.9 度，垂直 45.2 度
音声入力	ステレオマイクロホン
音声出力	スピーカー
内蔵センサー	赤外線方式距離センサー× 2
	加速度センサー
	振動センサー
入力センサー	頭タッチセンサー
	背中タッチセンサー
	あごセンサー
	肉球センサー

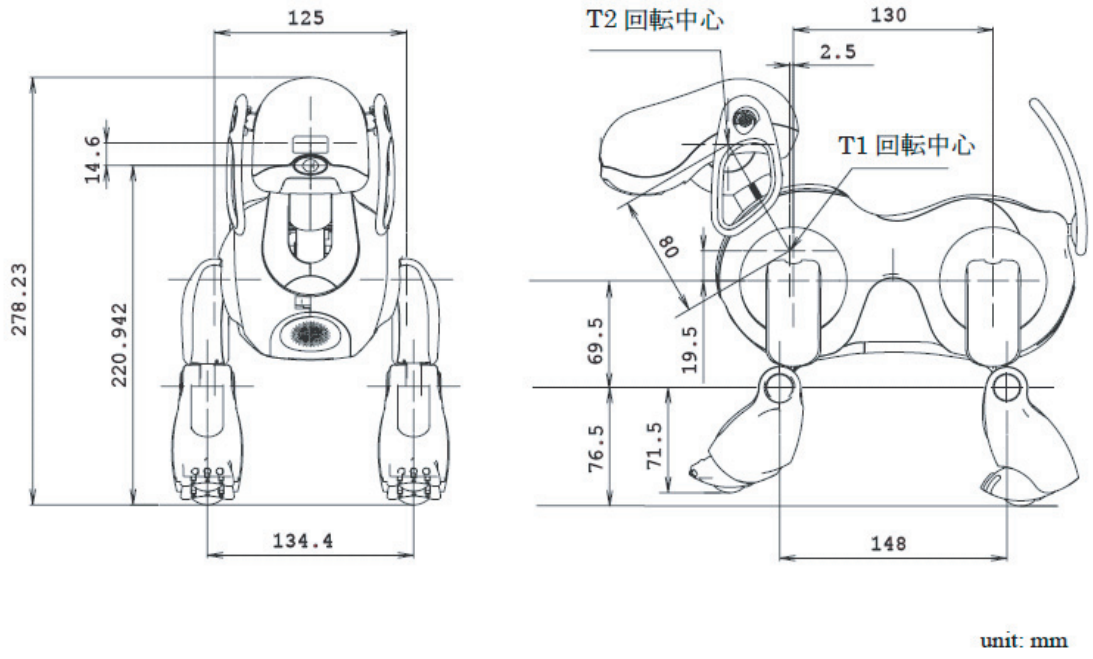


図 1.1: AIBO の外形

では行われている．キープアウェイサッカーとして知られるシミュレーションでパスワークの学習を行った [9] では，攻撃側と守備側にロボットがわかれ，攻撃側がボールをより長く味方でキープできるように学習する一方で，守備側はボールを早く奪取できるように学習をする．シミュレータではボールの摩擦や環境誤差を考慮する必要がない，さらに実ロボットと違いボールを扱うことが簡単である．また中型ロボットのキーパーの守備行動を学習を行った [10] では，シミュレータでキーパーの守備の学習を行い，そのデータを中型ロボットに用いて現実でのキーパーの守備行動の最適化を行った．全方位カメラをもつ中型ロボットとは違い四足ロボットでは視界が限られている，また歩行方法も足の関節角を制御しながら動くためにシミュレータとの動きの誤差が大きくなってしまう．このような事から，本論文ではシミュレータと四足ロボットとの環境の違いを近づけるため，拡張現実をもちいてキーパーの守備行動を学習するシステムを提案する．ロボットに関しては実ロボットを用いて，ボールや観測データに関してはシミュレータから与えられるものを用いる．このことでボールの摩擦や環境誤差がない中で実ロボットが動くことで，シミュレータから学習したデータを拡張現実でさらに学習を行う．

拡張現実を用いた実験はシミュレータと現実の環境の差をなくす事だけではなく，実

験にかかる労力を減らす．現実で実験を行う際には，まず人間ロボットを元の位置に戻して初期状態を作る．初期状態から学習を開始し，ロボットの行動に対してボールを投げて報酬を与える．さらにボールを拾って，ロボットを初期状態へ戻して，学習を再開する．この現実では労力のかかる実験を拡張現実では自動で行うことができる．これは実験自体の時間を短縮することにつながり，より多くの学習回数をこなすことができる．

本研究では学習を行うために拡張現実サッカーフィールドシステムを構築するのだが，現実世界の様々なタスクをコンピュータで補助するために，カメラとプロジェクタを使ったシステムの研究が行われており，例としてホワイトボードの拡張 [11] やビリヤードの補助 [12] がある．また，インタフェースにロボットを用いた研究として，既存インタフェースの延長としてロボットを用いることでより直感的な操作を可能とした研究 [13] [14] がある．その他に，ユーザが携帯型プロジェクタを用いてロボットを操作する研究 [15] もある．このアプリケーションでは，インタフェースとして完全に自律動作するロボットを用いることにより，インタフェースであるロボット自身がカメラとプロジェクタを通じて現実と仮想世界両方に積極的に関与するシステムを構築した．実験環境の応用例として，RoboCup サッカー四足リーグ [6] の環境を用いた．サッカーフィールド中にはこの環境を生かすのに必要な要素が十分含まれているため，この実験環境を試すには格好の対象であると言える．RoboCup において同じような環境を用いたリーグにフィジカルビジュアライゼーションリーグ [16] があるが，このリーグで使用されるロボットは制御を外部コンピュータが行うのに対し，本研究では各ロボットが完全に自律動作を行うという違いがある．

本論文は以下の様に構成される．第 2 章では AIBO の制御について述べる．第 3 章では本論文で用いた拡張現実サッカーフィールドシステムについて述べる．第 4 章では本論文で行った学習アルゴリズムを述べ，第 5 章では実験結果を述べる．最後に第 6 章では本論文の結論を述べる．

第2章

ロボット制御

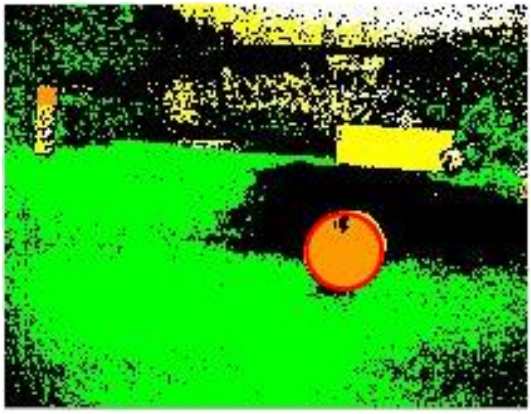
ここではロボット制御の仕組みについて簡単に説明をする．プログラムは JollyPochie として開発したプログラムを用いている．

ロボットを制御するまでの流れは次のようになっている．まずロボットが鼻先のカメラで画像を取得し，取得した画像に対して減色処理を行い代表色だけの画像を作る．その結果を図 2.1 に示す．図 2.1(a) がカメラで取得した原画像で，図 2.1(b) が減色処理を行い作成した画像である．減色処理をされた画像から，色や形状から物体の認識を行っている．図 2.1(a) の画像を処理した結果ボールであると認識されたものが図 2.1(b) の赤い丸がつけられている．さらに物体認識をした後に物体との相対位置から自己位置同定を行い，戦略プログラムにより行動を選択してロボットが動く．ロボットの歩行はさまざまな方向に動くことができ，戦略プログラムは動きたい方向を選択し歩く．またはゴールに向かってボールを動かすために予め用意されているモーションを用いてボールを蹴る．この一連の流れを繰り返す事によって，ロボットが画像から状況判断を行い自律で行動する．

2.0.1 画像処理

前述でも述べたようにボールの認識に関してはオレンジ色で丸い物体をボールとして認識する．ボール認識が終わると画像中のボールの大きさや位置などの認識結果とロボットの関節角から逆運動学を用いて相対距離と相対角度を計算する．この時ロボットの関節角がしてした角度になっていない場合や，地面に対する体の角度が通常の角度とずれる場合は逆運動学の計算に誤差が生じる．しかし AIBO に使用されるモーターはパワー

(a) 原画像．208 × 160 ピクセルの AIBO のカメラで取得した画像



(b) 減色処理をした画像．RoboCup のオブジェクトに用いられる色のみにした画像．

図 2.1: AIBO の画像

が弱いため指定した角度になっていないことが多く，そのため地面に対する体の角度がずれることも多い．また距離測定はボールの直径から計算しているのだが，AIBO のカメラは単眼であるため減色処理が正確にできなかったり，カメラの視界にボール全体が写っていない時などには誤差が生じやすい．これらの誤差を吸収するために観測値にパーティクルフィルターを用いてフィルタリングしているが，誤差が十分にはなくならない．

2.0.2 運動制御

観測だけでなく制御の面でも誤差が生まれてくる．シミュレータや中型ロボットが常に一定の距離を進めるのに対して，四足ロボットの動作は体バランスや前回の動きに影響されるために歩行が不安定である．その結果同じ歩行をしたとしても歩行距離や進行方向が指定した歩行と異なってくる．この歩行距離など AIBO の動きも観測時にフィードバックしているために動きで生じた誤差が，観測時の誤差を生む原因ともなる．

また AIBO が動きは一定の時間同じ行動をしなければならない制約がある．たとえば AIBO のガードモーションに関してはモーションを始まるとモーションが終わるまでは，他のモーションや歩行は実行されない．

また歩行に関しても制約がある．AIBO が前方向に進む際を例にとると，図 2.2 に示されるように AIBO の足の軌跡が示される．AIBO は常にどの方向にあるけばよいかとい

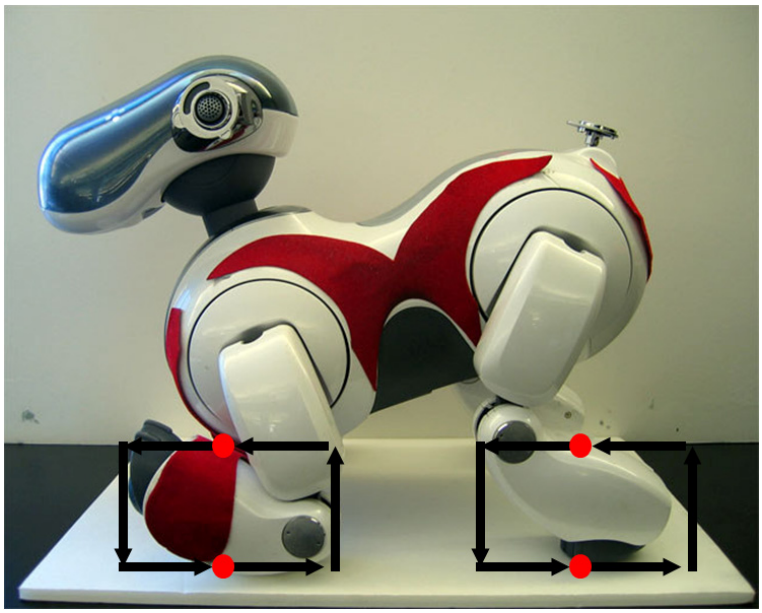


図 2.2: AIBO の歩行の軌跡．前進の時には黒の四角の角 4 点をエルミート補間した軌跡で足を動かし，赤い点のときのみ他の歩行に切り替えることができる．

うことを判断しているが，歩行中いつでも歩きを変えられる様にしてしまうと不安定な歩き方になってしまう．そこで図 2.2 で示された歩行の軌跡の中でも赤点で示されている 2 カ所の点でのみ歩行の切り替えができるようになっている．足が半周することに歩行の切り替えができるのだが，その間 4 から 5 ステップほどの時間がかかり，その間に何度歩行を変えようとしても最後に判断した歩行のみが次の歩行として選択される．

第3章

サッカーフィールドシステム

3.1 システム概要

この章では本論文で使用した拡張現実「サッカーフィールドシステム」について述べる．このシステムは2組のカメラ・プロジェクタとそれらを制御するコンピュータにより構成されている．コンピュータ内の制御・処理アプリケーションについては「認識プログラム」とロボットやボールの動きをシミュレートする「バーチャルアプリケーション」の2つのプログラムに分けられる．認識プログラムは天井カメラからの画像を元にロボットの認識を行なうプログラムである．バーチャルアプリケーションは認識プログラムの結果を元にロボットの実座標を計算して自己位置やシミュレータ内のボールの相対位置をロボットへ送信する．また実ロボットとシミュレータ内のボールとの衝突判定等を行なうプログラムである．システム全体の構成図は図 3.1 のようになる．カメラとプロジェクタは図 3.2 のように天井に設置した．カメラとプロジェクタの一組でコートの半分を撮影・投影できるため，二組でコート全体をカバーしている．カメラはPOINT GREYRESEARCH 社 [17] の Dragonfly2, プロジェクタは SANYO の LP-XL45 を使用した．また，フィールド上に投影した映像を十分視認できるようにするため，部屋の照明はシステム稼働時には使用せず，プロジェクタの光のみを用いる．また，プロジェクタは真下に投影しようとすると天井からコートまでの距離の関係でコート全体に投影できないため，図 3.3 のように反対側のコートに投影している．

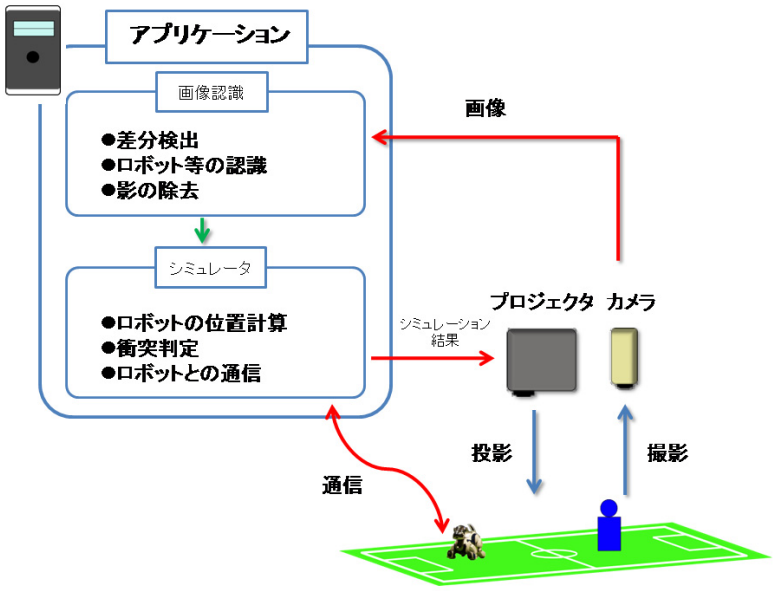


図 3.1: システム構成図. カメラによって実ロボットを認識する認識部と，認識結果を用いてシミュレーションする部分に分かれている

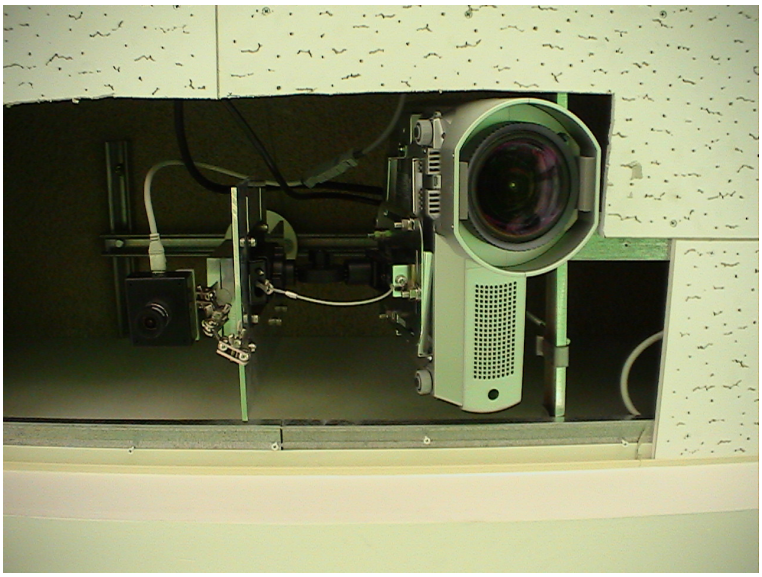
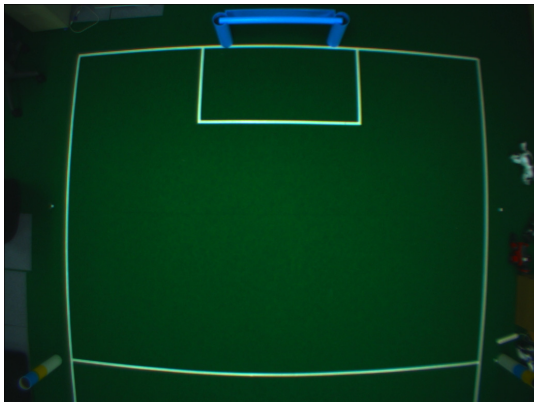


図 3.2: 天井の設置図. 左の黒い物体がカメラで，灰色の大きな物がプロジェクタである.

図 3.3: プロジェクタの投影方法．お互いに異なるコートの方を投影している．



(a) 青コート

(b) 黄色コート

図 3.4: 天井カメラの画像

3.2 認識プログラム

3.2.1 カメラ画像の取得

カメラ画像はPOINTGREYが提供するAPIを用いてPCに取り込んだ．取り込まれた画像はOpenCV [18]の画像フォーマットに変換し，OpenCVのライブラリを利用して画像処理や表示などを行なった．天井カメラは2つ設置しサッカーコートの半分ずつ撮影している．また，2つのカメラ画像の処理は独立しているため，マルチスレッドで処理を行なっている．

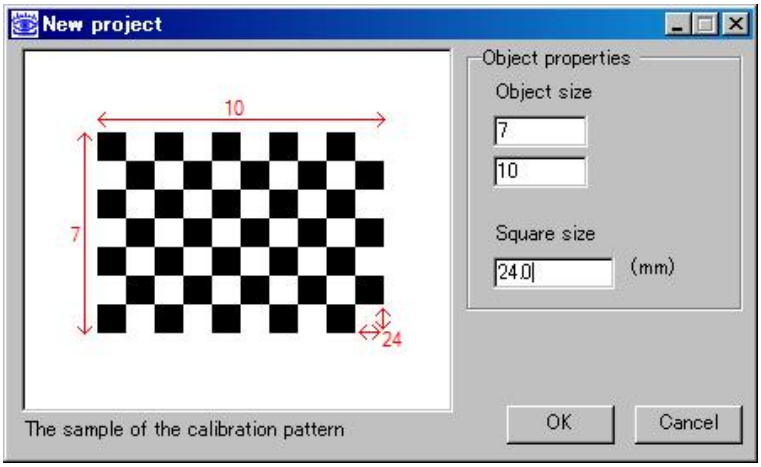


図 3.5: GML ツール

3.2.2 カメラの内部パラメータと外部パラメータの補正

本研究で用いたカメラのレンズは広角レンズなため，図 3.4 の白線部分に示されるように歪みが非常に大きい．このままでは外部パラメータ推定やロボットの認識に誤差が生じやすくなってしまうため，まず内部パラメータの補正を行った．GML C++ Camera Calibration Toolbox [19] を用いて，カメラの内部パラメータを 2 つのカメラそれぞれについて求めた．そして得られた結果と OpenCV を用いて，図 3.6 のように歪みを補正した画像を得た．内部パラメータを補正してレンズによる歪みを撮った後に外部パラメータを補正する．天井カメラで認識した各ロボットにフィールド上の 3 次元座標を送信する必要がある．そのためにはカメラ画像中の点が実世界の 3 次元座標のどこに対応するかを知らなければならない．カメラ画像中の 2 次元座標を実世界の 3 次元座標に変換するには外部パラメータを求め，カメラ画像中の座標に外部パラメータを求めて算出した射影行列をかければよい．本システムでは図 3.7 のようにフィールド上の 8 点を参照点とした．図にある 8 個の参照点を用いて外部パラメータを計算し，カメラ座標と実座標の変換を行なう．本論文で使用したシステムではカメラ，フィールド共に固定されているため，キャリブレーション専用のソフトを用いて必要に応じてキャリブレーションを行なった．

図 3.6: 歪みを補正した画像

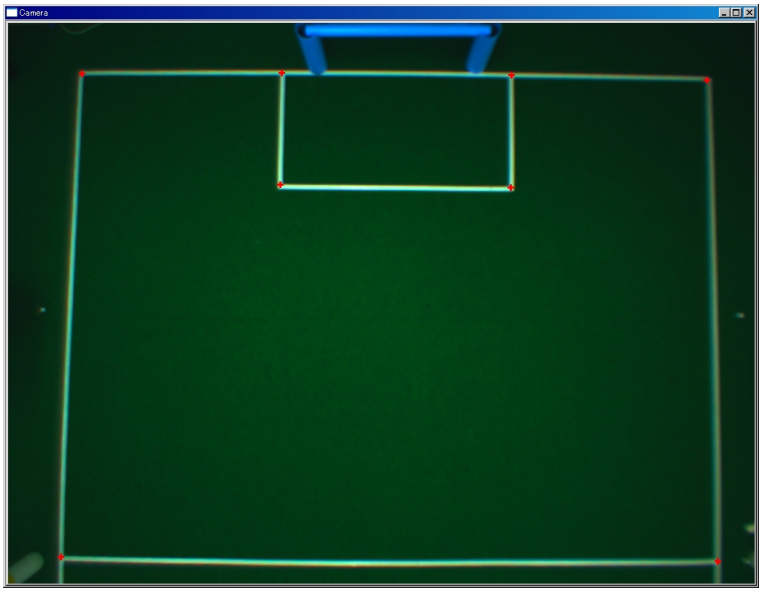


図 3.7: 外部パラメータの補正に用いる参照点

図 3.8: 認識の概要

3.2.3 認識方法

認識すべき対象物はフィールド上のオブジェクトであるが，オブジェクトの色は照明などの環境によって異なるため色を用いた認識は難しい．そこで本システムは画像中のオブジェクトの抽出方法として背景差分法 [20] を用いた．背景差分法とは，オブジェクトが何も映っていない背景の情報を記憶しておき，現在の画像の各ピクセルが背景に属しているか否かを何らかの基準で評価し，画像を前景と背景に分離する手法の事である．この手法を用いて認識すべきオブジェクト領域を示すマスク画像を作成した後は輪郭の認識を行ない各オブジェクトの認識を行なった．さらにロボットに位置と一緒に現在の向きも送信する必要がある．そこで認識した各ロボットに対し，最後にテンプレートマッチングを用いて向きの推定を行なった．認識の流れは図 3.8 のようになる．

3.2.4 背景差分の計算

最も単純な背景差分法は，抽出対象となる物体が画像中に入っていない状態の画像を背景画像として保持しておき，現在の画像の色情報と背景画像における色情報との差をピクセル毎に計算することで実現できる．しかし，一般に画像の場所によって明るさの揺らぎの大きさは異なる．そこで，本システムではピクセル毎のゆらぎが正規分布を成すと仮定し，背景画像を複数枚取得することでピクセル毎のゆらぎを個々に求めた．

正規分布は以下の式で表される．

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma}\mathbf{exp}(-\frac{(x-\mu)^2}{2\sigma^2})$$

μ は平均，σ^2 は分散となる．この正規分布を，背景画像を 100 枚集めることでピクセ

(a) 天井カメラの画像

(b) 背景画像

(c) 差分画像

図 3.9: 背景差分

ル毎に求め，画像のピクセルの値と，平均との差が標準偏差の定数倍以上であるならばそのピクセルは背景であるとした．ただし，入力画像は R,G,B の 3 チャンネルを持つが，これら一連の計算は 3 チャンネル別個に行なっている．当然閾値もチャンネル毎に異なる．そして 1 ピクセル中のチャンネルのどれか 1 つでも閾値を超えた場合，そのピクセルは背景と見なしている．この手法を用いた時の入力画像，背景画像，マスク画像を入力画像に適用した結果画像をそれぞれ図 3.9 に示す．これらから，背景差分法により適切に認識すべきオブジェクトが抽出できていることが確認できる．

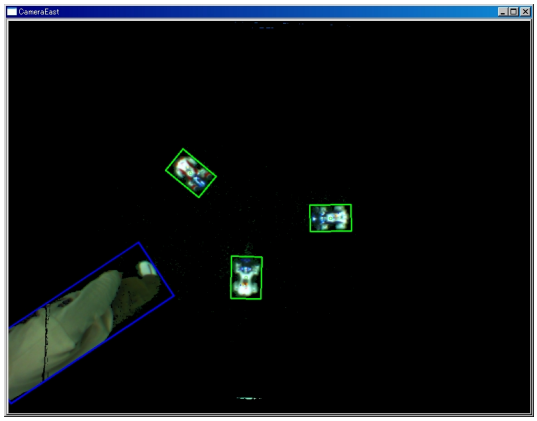


図 3.10: 認識結果．緑の四角はロボット，青の四角はそれ以外として認識した物体．

3.2.5 輪郭の抽出

背景差分の計算によって作成したマスク画像から輪郭を探索することで，ロボットの認識を行う．

背景差分の結果で認識されるものはロボット以外のフィールド上の物体も認識される，そこで輪郭長や輪郭内の面積の大きさに応じてロボットかその他のオブジェクトであるかを区別した．そして各輪郭に対し，その輪郭を囲む角度付きの最小矩形を OpenCV のライブラリを用いて求めた．角度は画像右を基準として算出している．認識結果は図 3.10 のようになる．緑の四角はロボット，青い四角はその他のオブジェクトとして認識されたことを表している．

3.2.6 テンプレートマッチングを用いた角度の推定

オブジェクトを囲む矩形がわかった，しかしロボットの角度を求めるためにはロボットの前後を同定する必要がある．ロボットの中心位置と角度は既にわかっているため，これらから図 3.11 のようにロボットの背中をテンプレートとしてロボットの前後に対してマッチングを行なった．マッチングを行なった結果は図 3.11 のようになり，矢印が認識したロボットの向きである．このことによりロボットの角度を求めることができた．

図 3.11: テンプレートマッチング．矢印でしめしたのがロボットの向きとして認識している．

3.2.7 結果の送信

認識結果を UDP 通信を用いてバーチャルアプリケーションに送信した．ただし，2 つのカメラ画像の処理は別スレッドで独立に行なわれているため，この UDP 送信処理も画像処理とはまた別スレッドで行い同期をとった．

3.3 仮想アプリケーション

認識プログラムの結果を元に実ロボットへのシミュレータ内の情報の送信や，現実と同様に衝突や摩擦などの物理シミュレートを行う．RoboCup に関係する４足ロボットを使ったシミュレータがいくつかあり，マイクロソフトがロボットの共通プラットホームとして公開している Microsoft Robotics Studio [21] や４足ロボットリーグに所属する日本のチーム Araibo [22] が公開している Haribote [23] などがある．この二つのうちソースコードまで公開されている，Haribote を参考に開発を行った．物理エンジンには ODE [24] を用い，画面の描画には OpenGL [25] を用いて実装している．またシミュレータの中には Araibo のプログラムで動く仮想ロボットも実装されている．この仮想ロボットはカメラベースで動いているが，実ロボットとは違ってシミュレータ内のオブジェクトを見ているため画像のノイズなど環境誤差などのない理想的な環境の中で動いている．我々のシミュレータは Haribote を参考に実装し，コートのサイズやオブジェクトの配置は RoboCup 4

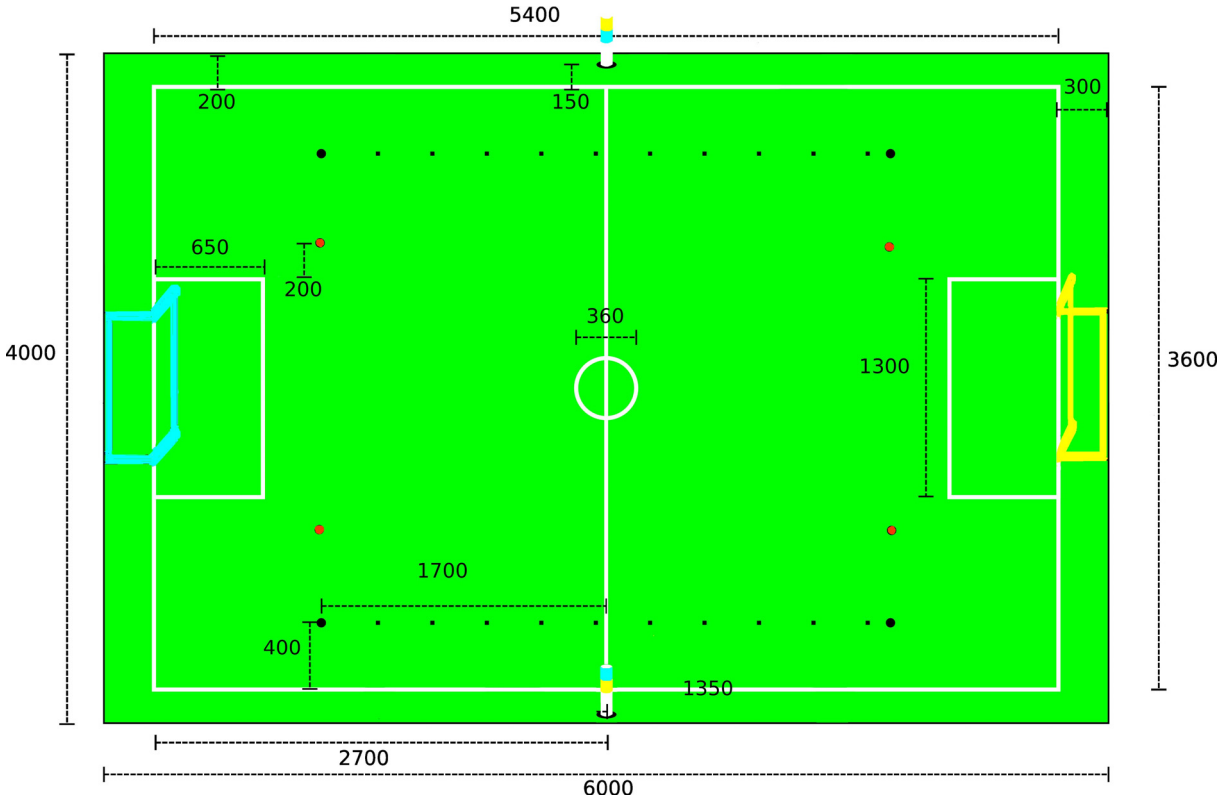


図 3.12: RoboCup の規定のフィールド

足リーグの 2007 年の公式ルールに記載されている規定のコート (図 3.12) 通りにアプリケーション内のフィールド (図 3.13) を作成した.

3.3.1 シミュレータ

現実のサッカーフィールドと同様のコートをシミュレータ内に作成し，フィールド上には仮想ボールと仮想ロボットが配置される．ODE により物理シミュレートを行い，仮想ロボットが仮想ボールに衝突すると仮想ボールに力が加わって動く．また現実の試合中はボールがフィールドから出たり，点が入ると審判である人間がボールを手で動かしているのだが，シミュレータでは自動的にボールを動かしている．

3.3.2 実ロボットへの情報送信

バーチャルアプリケーションでは認識プログラムが認識した実ロボットの情報を UDP によって受信する．受信した情報をもとに，現実の実ロボットの座標とシミュレータ内

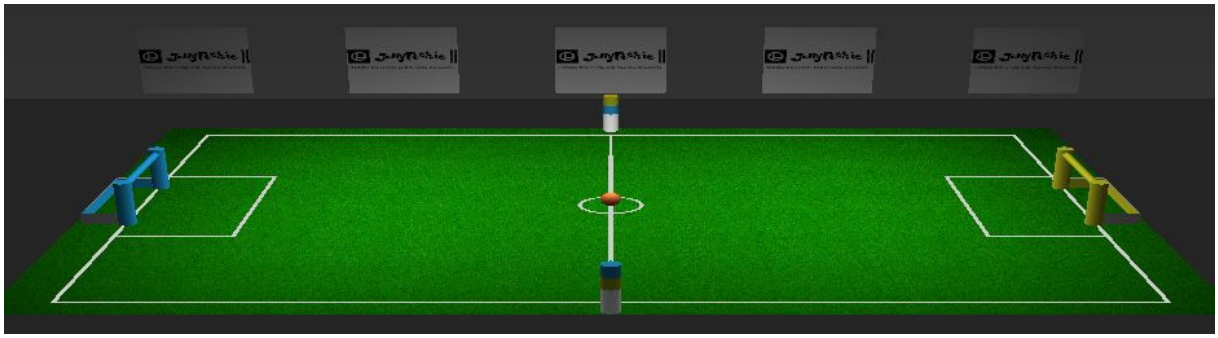


図 3.13: シミュレータのサッカーフィールド

の仮想ロボットの座標を比較して一番近いものを探す．シミュレータに仮装ロボットは一番座標の近かった実ロボットの認識結果を基に配置される．図 3.14 では実際にフィールドに実ロボットを置き，天井カメラで得た情報を元に実ロボットと同じ場所に仮想ロボットとして赤い直方体を配置した．これは実ロボットが移動すると，その動きに合わせてシミュレータ内の直方体も移動する．またバーチャルアプリケーションではシミュレータ内での自己位置，ボールの位置などを実ロボットに送信する．シミュレータ内のボールの相対位置や自己位置などを対応する実ロボットに UDP により送信する．送信した情報を実ロボットの認識した情報と置き換えることによって，実ロボットは通常の戦略プログラムによって動く．

3.3.3 現実とシミュレータの融合

シミュレータ内の直方体は実ロボットの動きに合わせて移動するだけではなく，ボールや他の直方体等に衝突する．この事により実ロボットがシミュレータ内のボールに衝突したときには，シミュレータ内でボールとロボットの衝突計算を行い，シミュレータ内のボールが実ロボットによって動かされた事になる．シミュレータは実ロボットへ位置情報やボールの位置を UDP 通信によって送信をしている．そうすることによって正確な情報を元に戦略スクリプトが動いている．実ロボットは実際の RoboCup の試合ではロボットについているカメラの画像を取得し，カメラに写ったオブジェクトを元に自己位置同定を行い，ボールをゴールに運ぼうと戦略にしたがって動く．しかしカメラのノイズのなどの環境誤差が入る．そのためロボット開発を行う際に環境誤差が入っているこ

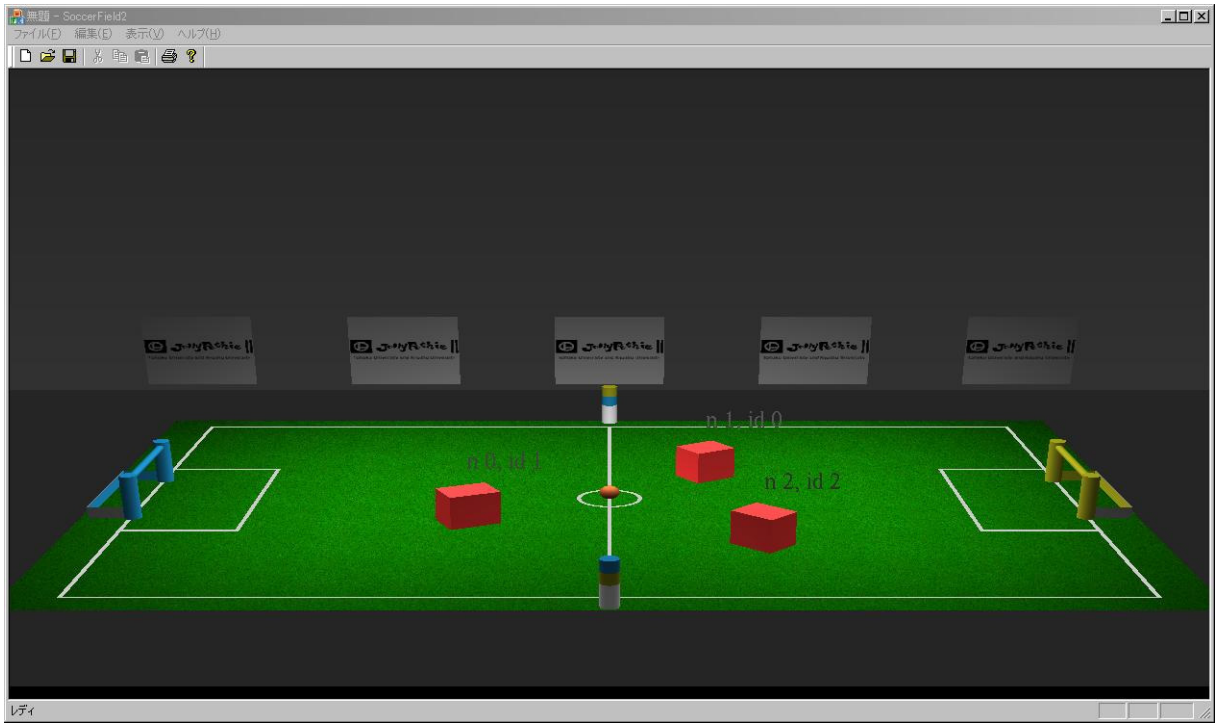
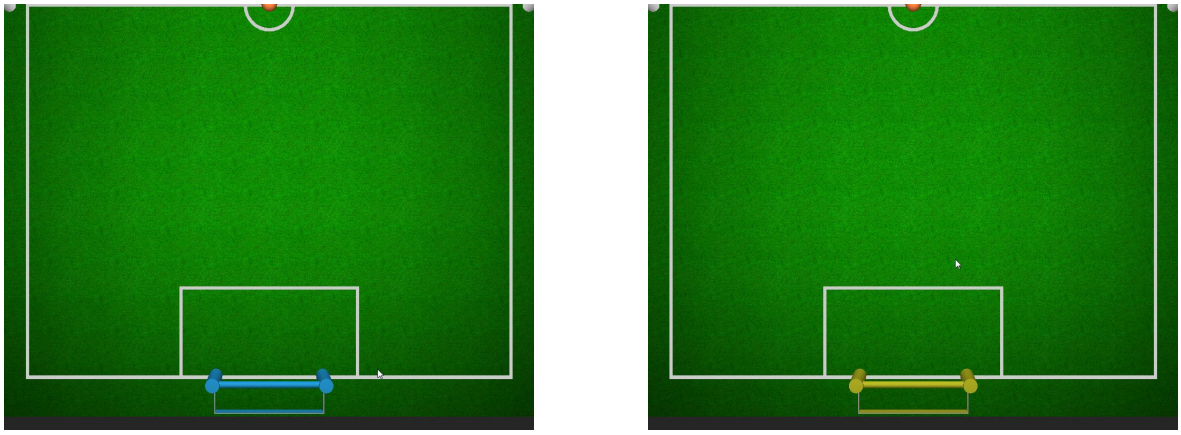


図 3.14: 実ロボットのシミュレータ内への配置．実ロボットを天井カメラで認識した結果をシミュレータ内に表示したもの．

とが原因でロボットの動作がおかしいのか，ロボットのプログラムが悪いのか評価しづらい．環境誤差のないシミュレータから正確な情報を受ける事によって，環境誤差を無視してプログラムの開発が進められる．その一例として，天井カメラで得た自己位置と，シミュレータ内でのボールとの相対位置を実ロボットへ送信する．実ロボットは受信した情報をもとに，戦略スクリプトを動かす．このことによって正確な位置情報が得られたと仮定して，実ロボットを動かせるため，位置情報の誤差と戦略スクリプトの性能との問題が切り分けられる．この他にもロボットのカメラに移る物体までの相対位置を実ロボットへ送信してやると，カメラのノイズがなかった時の自己位置同定モジュールが正確に動いているかどうかのデバッグができる．このようにシミュレータから送信する情報を変更することによって，実ロボットで動いている各モジュールの動作確認が行う事ができロボット開発を行う上で非常に有用である．



(a) 青コート　　(b) 赤コート

図 3.15: 上空からのフィールドの画像

3.3.4 プロジェクタからのシミュレータの投影

シミュレータ内の様子フィールドに設置したプロジェクタから投影することにより，フィールドを見るだけで実ロボットの様子もシミュレータ内の様子も見ることができる．プロジェクタで投影を行わなければ，実ロボットのいるフィールドを見て確認し，同時にパソコンの画面に映っているシミュレータ内の環境を観察しなければならない．そのためフィールドにシミュレータ内の様子を映し出すことによって，フィールドを見るだけで全ての情報が観察できるシステムを作った．天井に設置したプロジェクタはフィールドまでの距離が短いため，フィールド全体をプロジェクタで投影するのは不可能であった．そこで二台のプロジェクタを使って，それぞれがフィールドの半分をフィールドに投影する．そのためプロジェクタから投影する画像もシミュレータ内を半分に分けて，画像を作成しそれぞれのコートの上空から画像を取得する．実ロボットが動いている間も常に画像を更新し続けることで，図 3.16 のようにリアルタイムにボールの動きやロボットのデバッグ情報をプロジェクタから出力した．

図 3.16: プロジェクタからの投影．左に写っているのはシミュレータ内のロボットで，右に写っているのが実ロボットである．

第4章

学習

4.1 キーパーの学習

RoboCup では試合の勝敗がつかなかった場合にPK 戦が行われる．攻撃側のロボット一体と守備側のロボット一体を置き，攻撃側が点をいれるか，守備側が守りきってボールがコートから出るまで行われる．また攻撃側がペナルティエリアに入るとその時点で無得点で終わり，守備側がペナルティエリア外に出ると相手側に一点得点が入る．

本論文ではこの PK 戦におけるキーパーの行動を学習する．ボールの位置や速度，ゴールからのロボットの相対位置などの周りの状況から，どのような行動をとることが最適であるかを学習する．

4.2 学習方法

本論文では学習アルゴリズムに強化学習である Sarsa(λ)[26] を用いた．Sarsa(λ) は [9, 8] などで用いられており，キープアウェイや一次元でのボール捕捉などの学習をしている．強化学習は前提知識を必要としないので適切な報酬を与えてやるだけでよい．

強化学習の過程は状態と行動と報酬の系列で表され，

$$s_0, a_0, r_1,\ s_1, a_1, r_2,\ \ldots, s_i, a_i, r_{i+1},\ s_{i+1}, a_{i+1}, r_{i+2},\ \ldots,$$

はエージェントと環境の関係を表している．$s_t \in S$ はエージェントに与えられる時刻 $(t \geq 0)$ の環境の状態である．$a_t \in \mathcal{A}(s_t)$ は状態 s_t に対するエージェントの行動である．$\mathcal{A}(s_t)$ は状態 s_t でとりうる行動の集合である．行動に応じた報酬 $r_{t+1} \in \mathcal{R}$ を受け取り，状態 s_{t+1} へ遷移する．

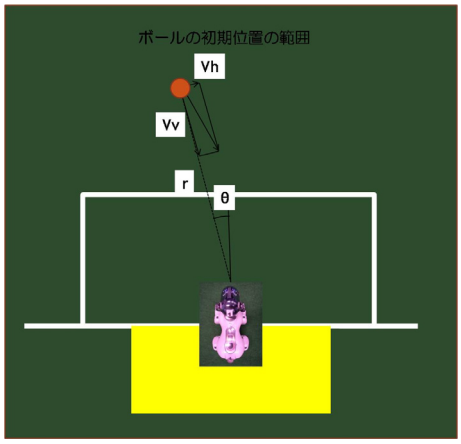


図 4.1: 学習モデル

学習の単位を１エピソードと呼び，ボールがペナルティスポットに置かれてから動き出したときに始まり，キーパーがボールをとめるか，もしくはゴールラインをわるまでを１エピソードとする．

ボールの相対位置，速度，さらにロボットのゴールからの位置を環境から与えられる状態とした．図 4.1 のようにボールの相対位置に関してはボールの距離 r と角度を ϕ，速度に関してはロボットとボールを結ぶ直線の水平成分を V_v，垂直成分を V_h とした．またゴールからのロボットの相対位置は直行座標系で X,Y とした．これら６個の観測データを状態として学習器に入力する．各変数の変域は r は $[0mm, 1000mm]$，ϕ は $[-\pi/2rad, \pi/2rad]$，V_v は $[-50mm, 0mm]$，V_h は $[-50mm, 50mm]$，X は $[-500mm, 1000mm]$，Y は $[-700mm, 700mm]$ とした．

AIBO はさまざまな歩行パターンをもっているが，本論文ではこのなかから縦，横，斜め方向に最高速度で移動する 8 パターンの歩行のみに限定した．これらの８パターン歩行に加え，その場に待機，ガードモーションの 10 の行動を学習の対象とする．

報酬は次の様に与える．

- エピソード終了時に得点をされていたら -10.0 の負の報酬を与える．
- 点を入れられていない場合はロボットの中心からとボールとの y 軸方向の距離に応じて距離が $0mm$ の時には報酬が 1.0 で，$225mm$ のときは -1.0 になるように報酬を与える．

- 点を入れられていない場合でキーパーがボールをガードして止めていたら，距離に応じた報酬にさらに 0.5 の報酬を加える．

図 4.2 は使用した自律学習のアルゴリズムである，エピソード的半マルコフ決定過程 Sarsa(λ) とタイルコーディング（CMCA として知られている）の組み合わせである．これは強化学習のよく用いられる組み合わせでキープアウェイ [9] でも使用されている．

行動 a 毎のタイルコーディングにより F_a には特徴集合が入っている．本論文では 6 次元のタイリングを用いていて，32 のタイリングと 414720 枚のタイルを使用している．また r のタイル幅を $250mm$，ϕ のタイル幅を $\pi/4rad$，V_v のタイル幅は $12.5mm$，V_h のタイル幅は $25mm$，X のタイル幅は $375mm$，Y のタイル幅は $350mm$ としている．$\vec{\theta}$ はパラメータ・ベクトルであり，学習の重み係数ともいう．Q_a は Q 値であり，F_a 毎の $\vec{\theta}$ の和によってあらわされている．通常 Q 値が最大となるような行動をとり，確率 0.001% の確率でランダムな行動をとる．適格度トレース $\vec{e}$ であり，過去の行動選択が現在の報酬を受けるべきかどうかの信頼性を保持している．λ は，適格度トレースのためのトレース減衰率であり，本論文ではトレース減衰率 $\lambda = 0.9$ とした．また学習率 $\alpha = 0.18$，割引率 $\gamma = 1.0$ としている．

```
1  while still not acquiring guarding skill do
2      s ← a state observed in the real environment;
3      forall a ∈ A(s) do
4          F_a ← set of tiles for a, s;
5          Q_a ← Σ_{i∈F_a} θ(i);
6      end
7      lastAction ← an optimal action selected by ε-greedy;
8      e⃗ ← 0;
9      forall i ∈ F_{lastAction} do e(i) ← 1;
10     reward ← 0;
11     while ball in the filed and the ball move do
12         do lastAction;
13         if lastAction = guard then
14             reward ← −0.02;
15         else
16             reward ← −0.0000001;
17         end
18         δ ← reward − Q_{lastAction};
19         s ← a state observed in the real environment;
20         forall a ∈ A(s) do
21             F_a ← set of tiles for a, s;
22             Q_a ← Σ_{i∈F_a} θ(i);
23         end
24         lastAction ← an optimal action selected by ε-greedy;
25         δ ← δ + Q_{lastAction};
26         θ⃗ ← θ⃗ + αδe⃗;
27         Q_{lastAction} ← Σ_{i∈F_{lastAction}} θ(i);
28         e⃗ ← λe⃗;
29         if player acting in state s then
30             forall a ∈ A(s) s.t. a ≠ lastAction do
31                 forall i ∈ F_a do e(i) ← 0;
32             end
33             forall i ∈ F_{lastAction} do e(i) ← 1;
34         end
35     end
36     if stop the ball then
37         if lastAction = guard then
38             reward ← 1 − dist/112 + 0.5;
39         else
40             reward ← 1 − dist/112;
41         end
42     else
43         if lost the goal then
44             reward ← −10.0;
45         else
46             reward ← 1 − dist/112 + 0.5;
47         end
48     end
49     δ ← reward − Q_{lastAction};
50     θ⃗ ← θ⃗ + αδe⃗;
51 end
```

図 4.2: 自律学習のアルゴリズム.

第5章

実験

5.1 概要

RoboCup四足リーグのルールでPK戦のときは図x5.1のようにボールはゴールの$1m$前に置かれ，ゴール前にキーパーが，攻撃側がボールの前に配置される．PK戦が開始されると攻撃側はペナルティエリアに入らないようにボールを動かして，ゴールの隙間を狙ってシュートする．この時キーパーはペナルティエリアから出ない範囲で，攻撃側の動きに合わせてシュートコースを消すためにポジションをとり，シュートされたボールに対しては手を広げてボールを止めるのが一般的である．

本論文ではキーパーの学習を行うため，攻撃側のロボットは必要ないため図のようにキーパーのみをゴール前に，ボールをゴールから$1m$離れた状態を初期状態として学習を行う．初期配置から攻撃側がボールをドリブルしてシュートする事を再現するため，エピソードが始まるとペナルティスポットから離れたところにランダムにシュート地点を決めてやり，シュート地点に向かってボールが動きだす．シュート地点に達するとボールがゴールに向かって動き出す．キーパーはボールの位置や速度を見ながら最適な行動はなにかを学習する．ボールはゴール中心からゴール幅から$150mm$離れたところまでの間に向かってシュートされる．

学習はシミュレータ，拡張現実の二段階で行う．シミュレータで学習した結果を用いて，拡張現実で学習する．ノイズのない理想的な状態を学習器に入力した結果を拡張現実で学習する．現実のようにノイズが多く学習することができないが，段階的にノイズを状態にいれることで理想環境での学習結果をノイズの含まれた環境でも最適な行動が選択できるデータを作成できるようにする．

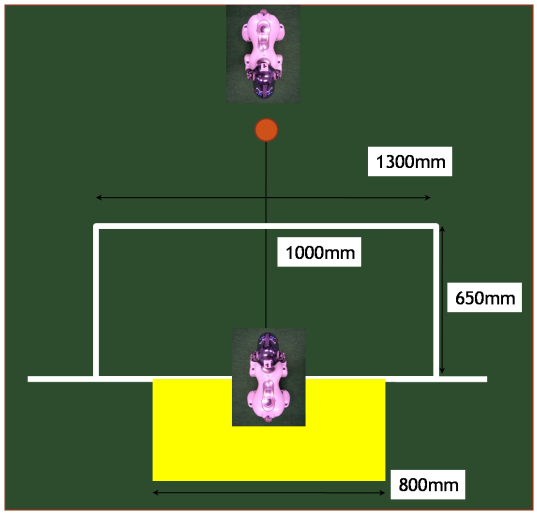


図 5.1: PK 戦での初期状態

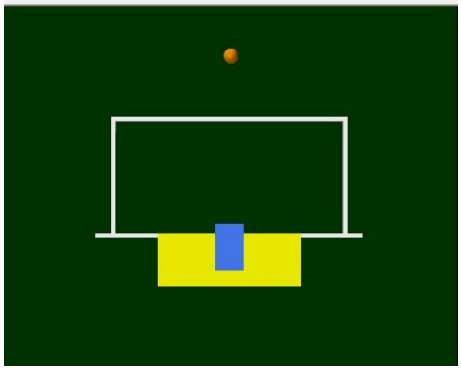


図 5.2: シミュレータ

5.1.1 シミュレータでの実験

拡張現実を用いて学習を行う前により高速に実験するためのシミュレータを使って学習を行う．そのために簡単なシミュレータを作成した．

図 5.3 のように AIBO を直方体の物体とし，黄色の四角で表示したゴールの前に配置する．白線はペンルティラインを示している．AIBO やフィールド上の物体は実際の物と同じサイズにした．

シミュレータを用いてロボットが理想的に動く環境で実験を行った．ロボット，ボールともにシミュレータ内の物を使って学習を行う．学習のアルゴリズムに関しては第 4.1 章で述べた通りである．状態として学習器に入力するボールの相対位置，速度，ロボットの位置などは全てシミュレータ内で計算する．行動もロボットの移動もシミュレータ

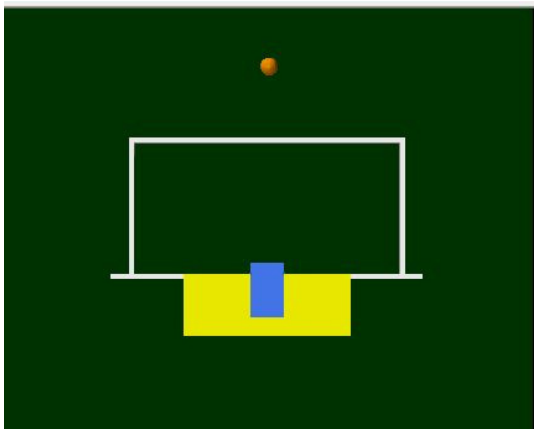
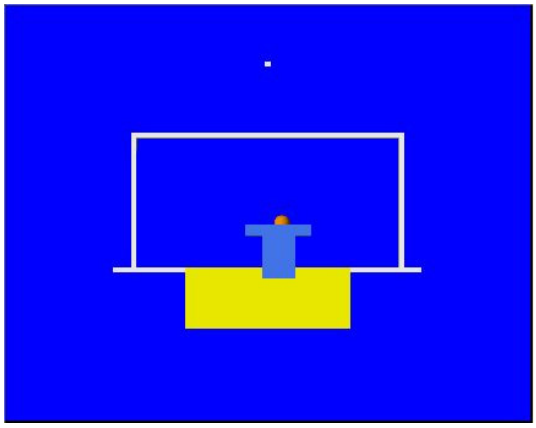


図 5.3: シミュレータでの学習の初期位置．ロボットの初期位置が原点以外の場所になっている．

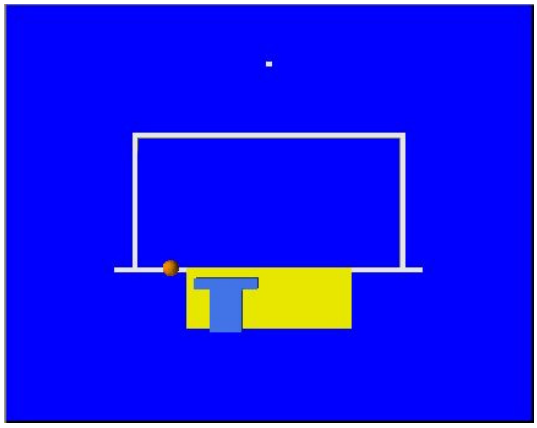
内で計算した．シミュレータでの学習の初期位置は図 5.3 である．ここから図 5.4 に示したガードの成功，ボールがフィールドからでる，キーパーがペナルティエリアから出る，点を入れられるの 4 条件のどれかの状態になるまでを 1 エピソードとして学習を行う．

パラメータの変化による影響

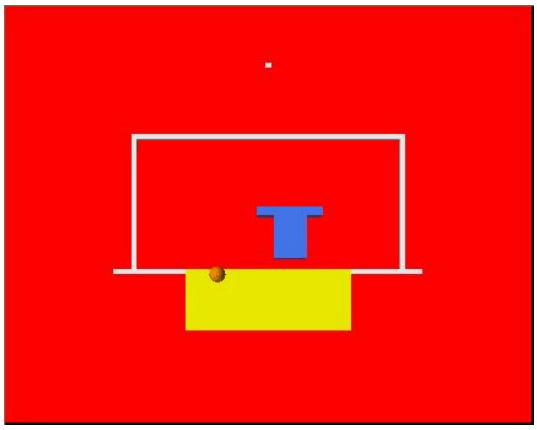
本研究では学習率を 0.18，ランダムに行動を選択する確率 $\epsilon = 0.001$，トレース減衰率 $\lambda = 0.9$ として実験を行った．これらのパラメータが変化すると学習の進み方がどのように変化するのかを観察する．3 つのパラメータをそれぞれ変化させて 2000 エピソード学習を行い 100 エピソードごとの成功率をグラフにした．ランダムに行動を選択する確率 $\epsilon = 0.001$，トレース減衰率 $\lambda = 0.9$ として学習率を 0.18，0.3，0.6 と変化させた結果を図 5.5 に示す．学習率を 0.18 としてトレース減衰率 $\lambda = 0.9$ としてランダムに行動を選択する確率を 0.0001，0.001，0.01 と変化させた結果を図 5.6 に示す．学習率を 0.18 としてランダムに行動を選択する確率 $\epsilon = 0.001$，トレース減衰率を 0.3，0.7，0.9 と変化させた結果を図 5.7 に示す．図 5.5，図 5.6 より学習率，ランダムに行動選択する確率は値を変化させても，学習にはあまり影響しないがことが分かる．一方で図 5.7 よりトレース減衰率については 1 に近づくほどより早く学習することが分かる．これは受けた報酬について過去の行動についても反映させた方が，学習の進みが早くなることが分かる．



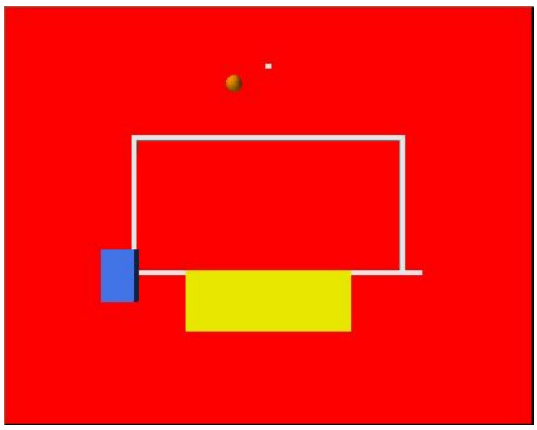
(a) ガードの成功



(b) ボールがフィールドから出た



(c) 点を入れられる



(d) キーパーがペナルティエリアから出る

図 5.4: エピソードの終了条件

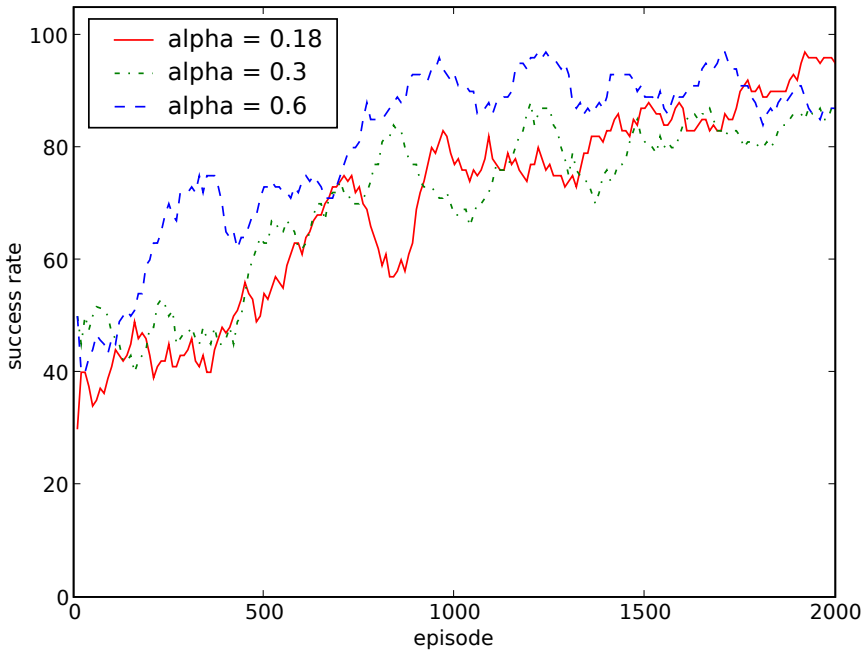


図 5.5: 学習率による変化

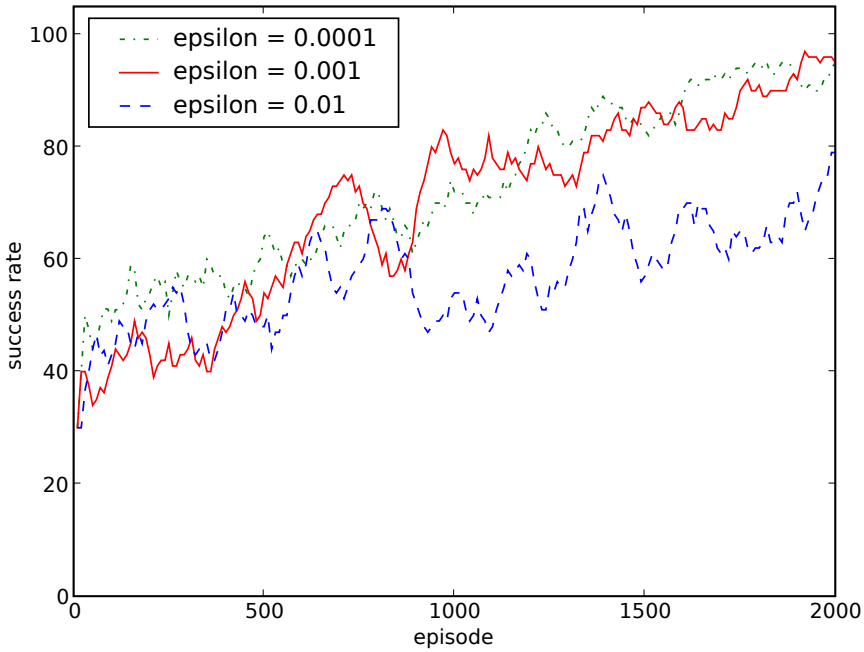


図 5.6: ランダム性による変化

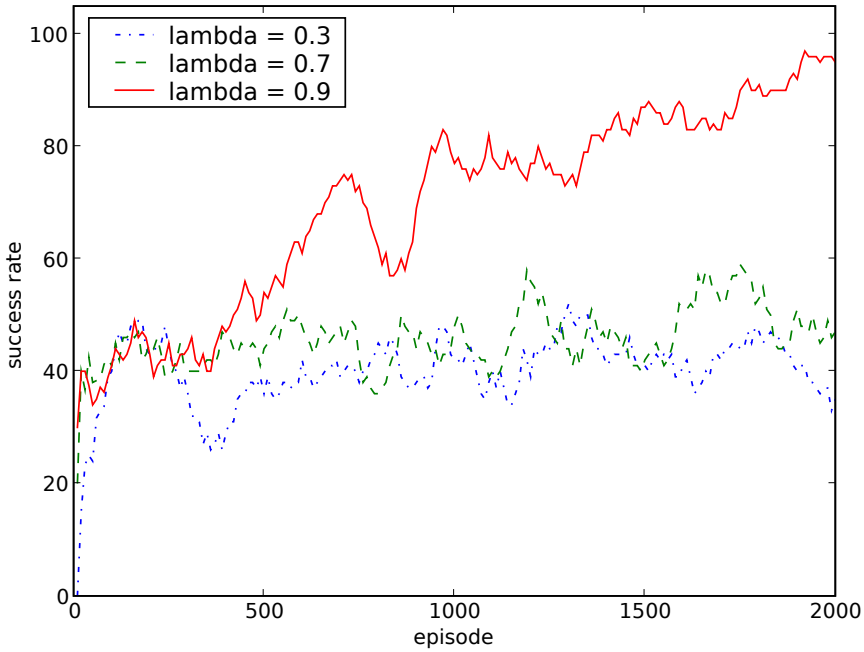


図 5.7: トレース減衰率による変化

シミュレータでの実験結果

シミュレータを用いて 2000 エピソード学習を行った実験結果を図 5.8 に示す．まず一番上のグラフは成功率を示すもので横軸はエピソードで，縦軸は 2000 エピソードを 20 エピソードずつに区切り過去 100 エピソードの成功率をプロットした．1000 エピソードを過ぎた辺りで 100% に近い成功率になった．次に二番目のグラフはエピソード中に何回ガードをしていたかを計測して，成功率と同様 2000 エピソードを 20 ずつに区切り，過去 100 エピソードのガードした回数の平均をプロットした．手を広げていればをしていればゴールの半分をガードできてしまうが，積極的にボールに近づいてからボールを止められるぎりぎりのタイミングでガードをすることが望ましいためガードした回数をプロットした．グラフからはエピソードが経過するとともにガード時間の平均が少なくなっていることから積極的にボールを止めに行く事を学習していることがわかる．最後に三番目のグラフはボールがロボットの横を通過した時のボールの横方向の距離を 100 エピソードの平均をとってプロットしてたものだ．これもエピソードが経過するとともにロボットとボールとの距離が少なくなっていることが分かる．また学習後のデータを用い

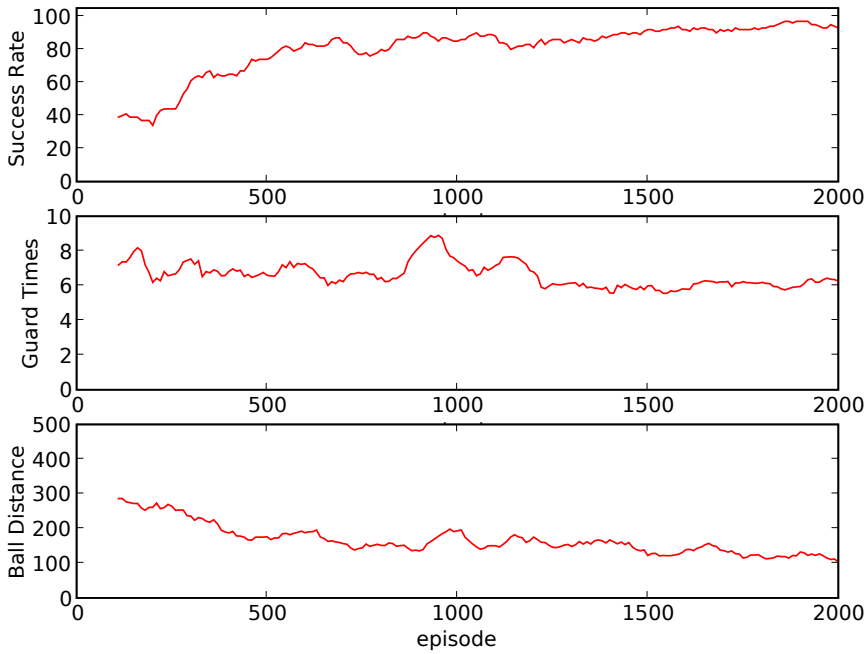


図 5.8: シミュレータでの学習結果

て，一定のボールの相対位置と速度に対する行動の選択をプロットした．図 5.9 に示すようにロボットが原点にいるとき，ボールの相対位置と速度を入力したときにロボットがどのような行動をしたかを．行動毎に分けてプロットしてどのような状態の時に行動が選択されるかを見る．図 5.9 ように横軸はロボットの X 座標，縦軸が Y 座標で学習器に入力するボールの相対位置と速度はシミュレータと同様にした．図 5.10 は学習の初期段階で 2 エピソード学習したデータにサンプルを入力して，一定のボールに対して，さまざまな位置でロボットがどの行動を選ぶかを示した．図 5.11 は学習終了後のもので 2000 エピソード学習したデータに関しても同様にプロットした．

グラフの横軸は X 軸，縦軸が Y 軸である．黒の四角はペナルティラインを示していて，青く塗りつぶされている四角がゴールである．それぞれのグラフに一つだけオレンジ色の丸が描かれているのがボールの位置で，そこから出ている黄色の線がボールの速度ベクトルを示している．この様なボールの位置，速度に対してロボットの位置を変えて，どの行動が選択されるかを示した．黒い矢印の始点や点または×印の位置がロボットの位置をしてしている．矢印がかかれている時には，矢印の始点の位置では矢印の方向に移

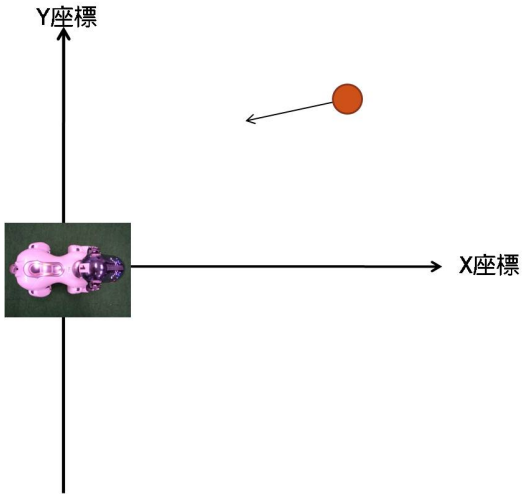


図 5.9: ボールに対する行動選択模式図

動する行動を選択したことを示している．また×印の場所では待機を，赤点の場所ではガードを選択したことを示している．学習の前後で比較してみると，未学習時はまったくランダムに行動が選択されているのに対して，学習後にはボールの軌道から $200mm$ 前後離れた場所では，ボールの軌道上の移動する行動を選択乎している．その中でも軌道上でボールに近い場所ではガードを選択している．一方その他の場所ではボールの軌道上による行動は選択することができず，学習できていないことがわかる．

5.1.2 拡張現実での実験

シミュレータを用いた学習の結果を用いて，拡張現実での学習を行った．ロボットはAIBO を使用し，ボールに関しては第 3 章のサッカーフィールドシステムのシミュレータ内の仮想ボールを使用した．サッカーフィールドシステムの学習での初期位置を図 5.12 に示す．第 3 章で述べた天井カメラのロボットの認識方法では場所によってロボットの認識の精度が異なってくる．ロボットがカメラの真下以外の場所では，ロボットの側面が見えロボットの中心が見えるために天井カメラによるロボットの位置認識に誤差が生じる．よってロボットの位置認識の精度が高い状態で実験を行うためににロボットの初期位置を天井カメラの真下にした．ペナルティエリアの位置とボールの初期位置はロボットの初期位置に合わせて移動した．ゴールに関してはシミュレータ内の表示は変更していないがゴールに入ったかどうかの判定の際には，ロボットの初期位置より後ろに行っ

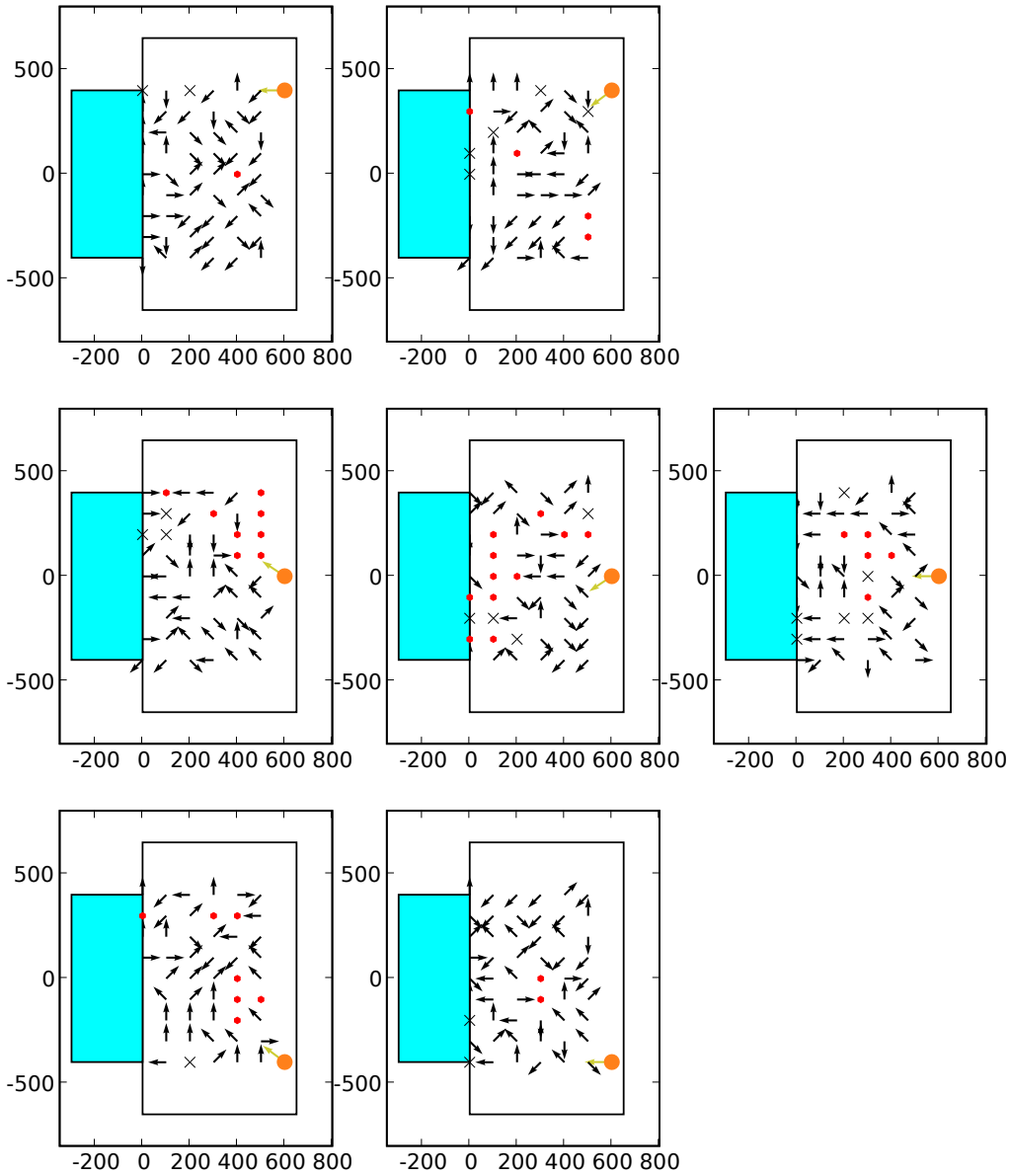


図 5.10: 2 エピソード後のボールに対する行動選択．黒の四角はペナルティラインを示していて，青く塗りつぶされている四角がゴールである．オレンジ色の丸がボールの位置で，ボールから出ている黄色の線がボールの速度ベクトルを示している．黒い矢印の始点や点または×印の位置がロボットの位置をしてしている．矢印がかかれている時には，矢印の始点の位置では矢印の方向に移動する行動を選択したことを示している．また×印の場所では待機を，赤点の場所ではガードを選択したことを示している．

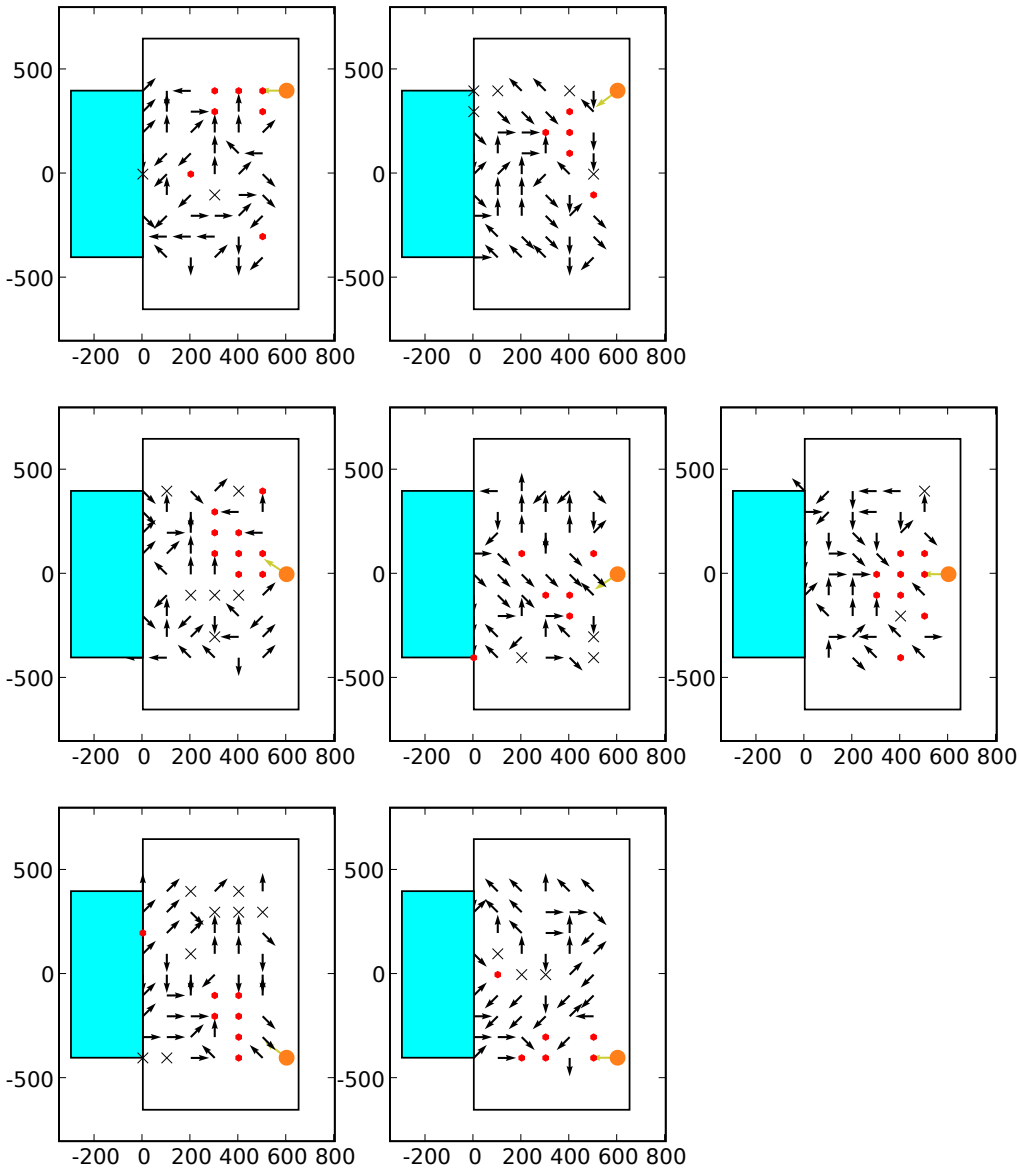


図 5.11: 2000 エピソード後のボールに対する行動選択

(a) シミュレータ内での仮想ロボットの初期位置

(b) 実ロボットのフィールドでの初期位置

図 5.12: サッカーシステムでの学習の初期位置

た時のボールの位置を見てゴール幅に入っていれば点を入れられたことにした．

まずは拡張現実のシステムのみを用いて，キーパーの学習を行う．学習データのないところから学習を行った結果を図 5.13 に示す．学習初期では約 40% の成功率で，200 エピソード後には約 60% の成功率となっている．ガード回数，ボールとの距離ともに学習が進んでいる様子が分かる．

ここで実ロボットで行う学習時間を短くするために，学習初期段階をシミュレータで学習しその結果を用いて拡張現実で学習を行う．実験結果を図 5.14 に示す．図 5.14 の成功率を見ると学習初期段階では成功率がほぼ60% と低いが，徐々に成功率が上がってきている．学習初期段階ではシミュレータと拡張現実との間で環境の差があるために，シミュレータの学習結果がそのまま適応することができていないが，シミュレータでの学習後に学習を行っているためにノイズが入っている拡張現実での学習としては学習の進行が早い．またガード時間，終了時のボールとの距離のグラフを見ても，シミュレータと同様に学習が進行していることがわかる．

5.2 初期位置の変更

拡張現実での実験を行った結果，学習の初期段階でのガードの成功率は60% しかない．これはシミュレータと拡張現実の環境誤差があるためと考えた．拡張現実ではエピソードが終了すると再び学習を始めるために，初期位置に戻るために移動する．この時に原点から半径 200mm 以内の位置につくと初期位置に戻ったとして学習を再開する．つまり

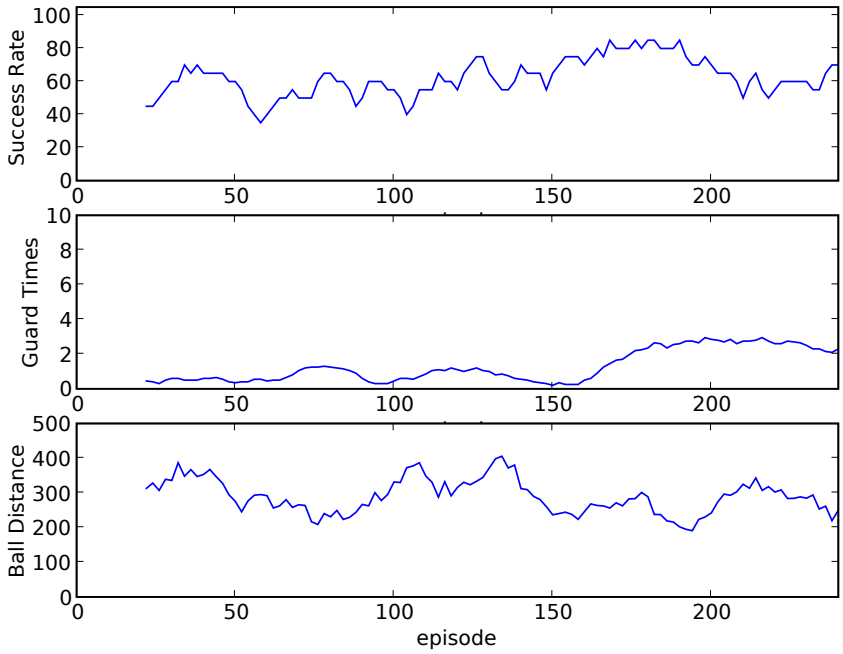


図 5.13: 学習初期段階から拡張現実で学習をした学習結果

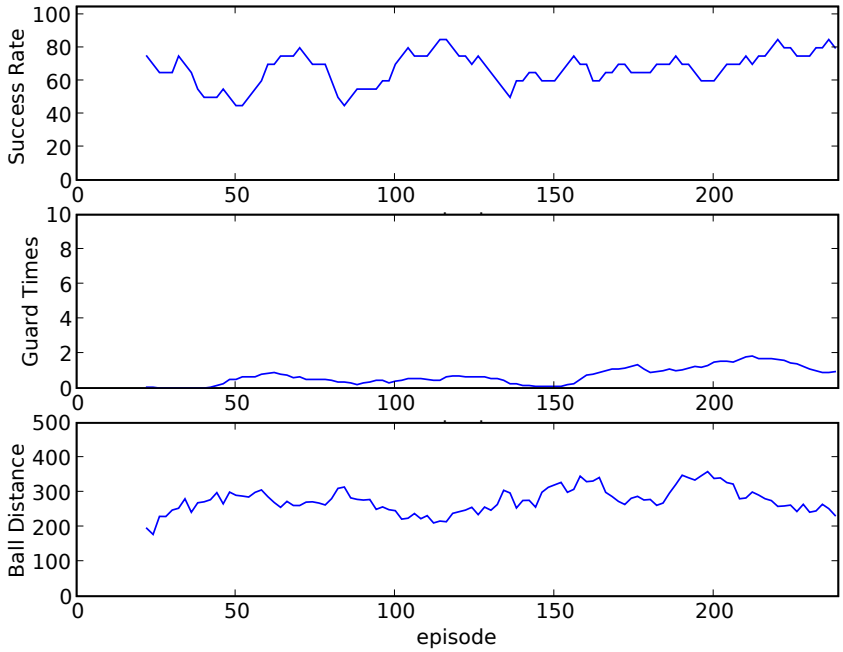


図 5.14: シミュレータでの学習結果を用いての拡張現実での学習結果

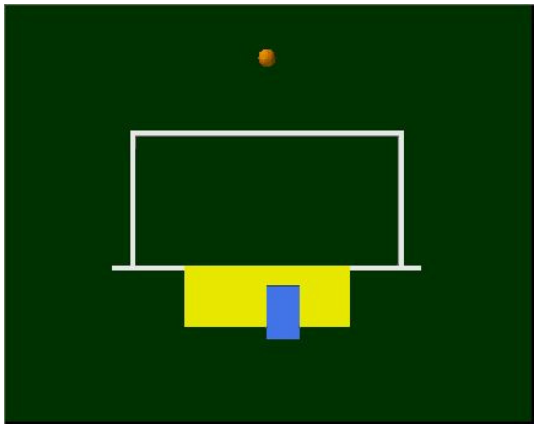


図 5.15: 初期位置を変更したシミュレータの初期状態

シミュレータでは原点からエピソードが始まるのに対して，拡張現実は原点から離れた場所からエピソードが始まる．この初期状態の違いはシミュレータでの学習が終了した時点で，拡張現実での初期状態であるボールがペナルティスポットにありロボットが原点から少し離れた状態が未学習であるということになる．そのためこの初期状態の違いをなくし初期状態を近づけるために，図 5.15 のようにシミュレータのロボットの初期位置も原点から離れた場所から始まるように変更をした．このときロボットの初期位置は原点から X 軸，Y 軸ともに $0mm$ から $150mm$ 離れた場所にランダムな位置にエピソードごとに違う場所になる．このようにシミュレータの初期状態を変えることで拡張現実での初期状態も学習済みのデータを作成して拡張現実で引き続き実験を行い学習の初期段階でもある程度の成功率が得られるデータで学習を行いたいと考えた．その結果を図 5.16 に示す．このグラフから学習の進み方は図 5.8 が 1000 エピソードで成功率が 100% に近づいたのに比べて，図 5.16 では学習が遅くなり成功率が 100% になるには 2000 エピソードかかったことが分かる．一方でガードの回数やボールの距離に関しては違いがない．

この初期位置を変更したシミュレータでの学習結果を用いて，拡張現実で学習を行った結果を図 5.17 に示す．図 5.8 と比較すると拡張現実での学習初期では，成功率が高いが，成功率が安定して向上せず大きく上下しながら学習が進行している．

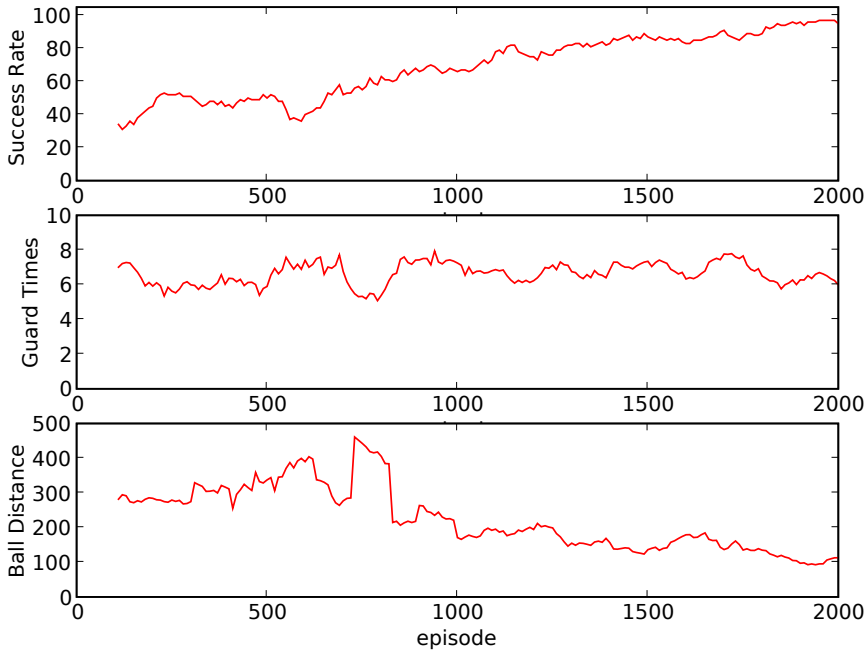


図 5.16: 初期位置を変更したシミュレータでの学習

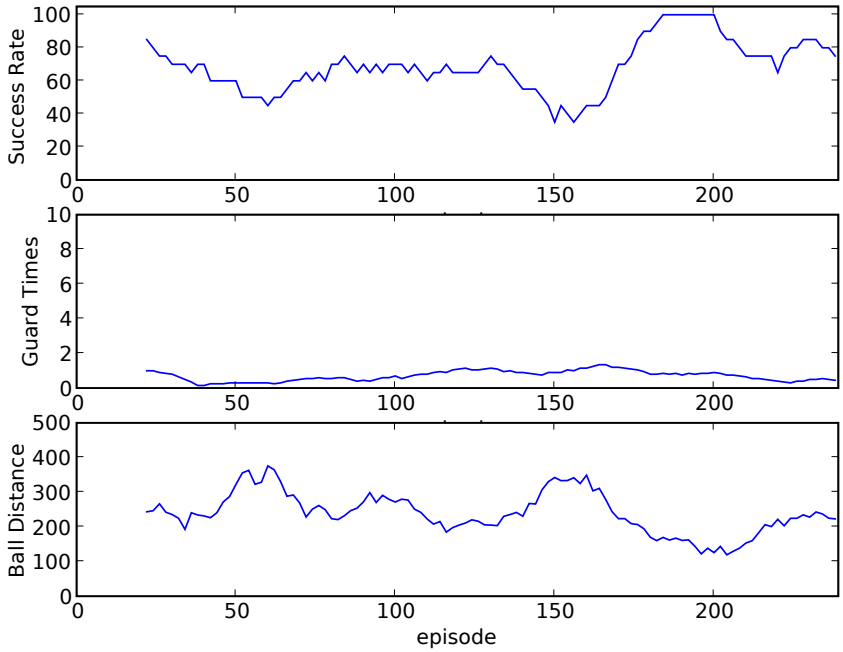


図 5.17: 初期位置を変更したシミュレータの学習結果を用いての拡張現実での学習

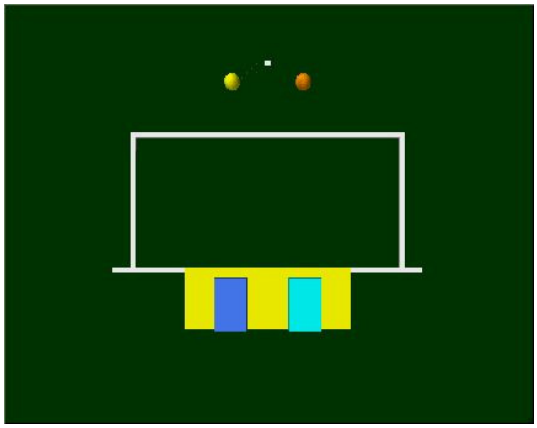


図 5.18: 左右対称にした状態．黄色が左右対称にしたボールで，水色が左右対称にしたロボットの位置である．

5.3 対称性を利用した学習

第 5.2 節で述べたようにシミュレータによるキーパーの学習ができたのだが，学習の効率を上げるための実験を行った．通常の学習の際に学習機は観測値を元にした状態に対して，行動を選択し，報酬を受ける．キーパーの学習では，学習機に入力するボールの相対位置，速度，ロボットのゴールからの位置などの状態，さらにロボットの行動，これら全てがロボットの X 軸に対して対称性がある．そこでこの対称性を利用して，ボールがロボットの右側にある時には図 5.18 のように観測された状態，選択した行動をロボットの X 軸に対して左右対称にして通常の学習時と同じ報酬を与えた．その結果を図 5.19 に示す．図 5.16 の通常の学習の結果と比較していく．通常の学習では 2000 エピソード付近で学習率 100% に成功率になっているのに対して対称性を利用した学習では 700 エピソード付近で学習率 100% に成功率になっていることが分かる．学習初期段階でより効率学習している事が分かる．またガード回数，ボールの距離ともに，通常の学習よりも対称性を利用した学習の方がより早く学習している事が分かる．また 2000 エピソード後のガード回数，ボールとの距離についても対称性を利用した学習の方がよりよい値になっていることが分かる．

また通常の学習と同様に学習後のデータにボールの相対位置，速度を入力して，ロボットが場所によってどのような行動をとるかをプロットものを図 5.20 に示した．ボールがロボットの右側にある場合は左右対称にして学習器に入力をしているため，ボールが常

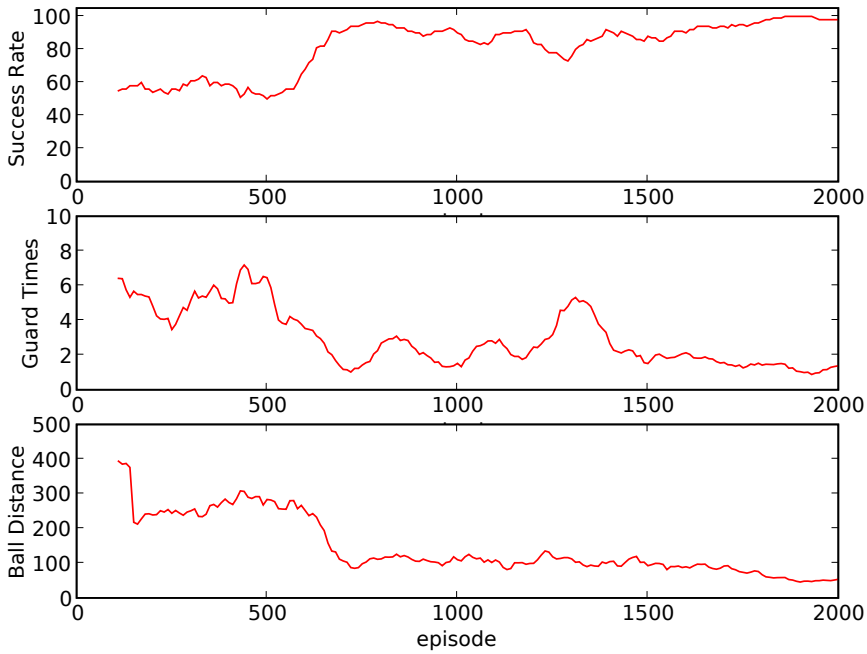


図 5.19: 対称性を利用したシミュレータでの学習の結果

に右側にある下段のグラフはすべて場所が未学習である．そのため選択した行動がほぼランダムになっている．中段以上のグラフについては，対称性を利用した学習行っているため通常の倍の学習をしていることになる．図 5.11 ではボールの軌道上の付近はすべてボールの軌道上に移動する行動を選択していたが，図 5.20 ではボールの軌道上付近でさらにボールが左にある場所に限りボールの軌道上に移動する行動を選択していることがわかる．続いてシミュレータの結果を用いて，拡張現実での学習を行った．図 5.17 と比較すると，安定的に学習が進んでいることが分かる．

5.4 ノイズの入った状況での学習

拡張現実や実ロボットで実験を行う際には観測値に誤差が生じる．そこでシミュレータの観測値に対してノイズを入れて実験を行うことにより，拡張現実や実ロボットでの学習とシミュレータの学習の差を少なくする．正確にシミュレートするためには現実でのノイズがどのようなものか正確に計算を行い観測値にノイズを入れるべきなのだが，本論文ではシミュレーションは拡張現実での初期学習を省略するためで正確なシミュレー

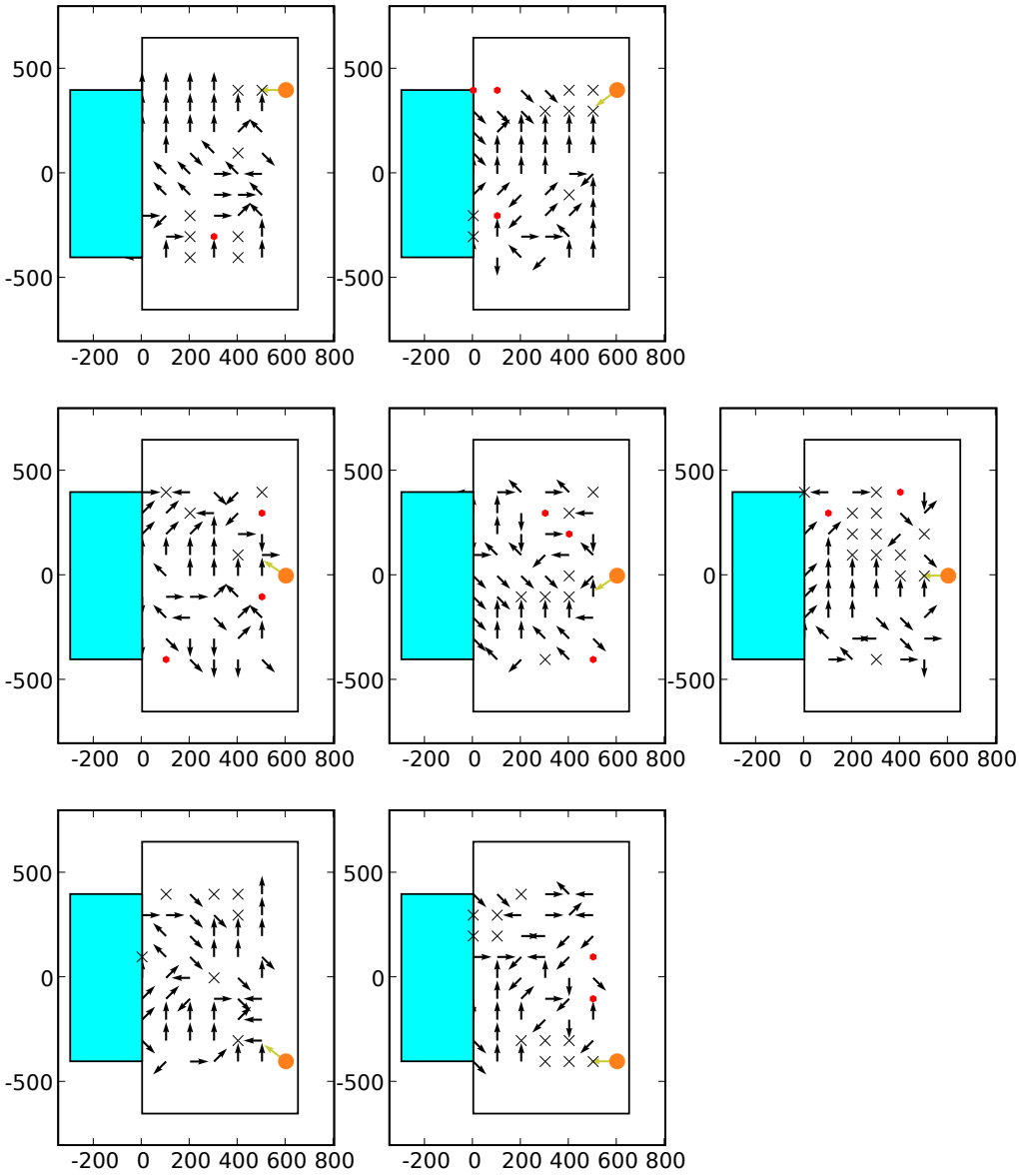


図 5.20: 対称性を利用した学習のボールに対する行動選択

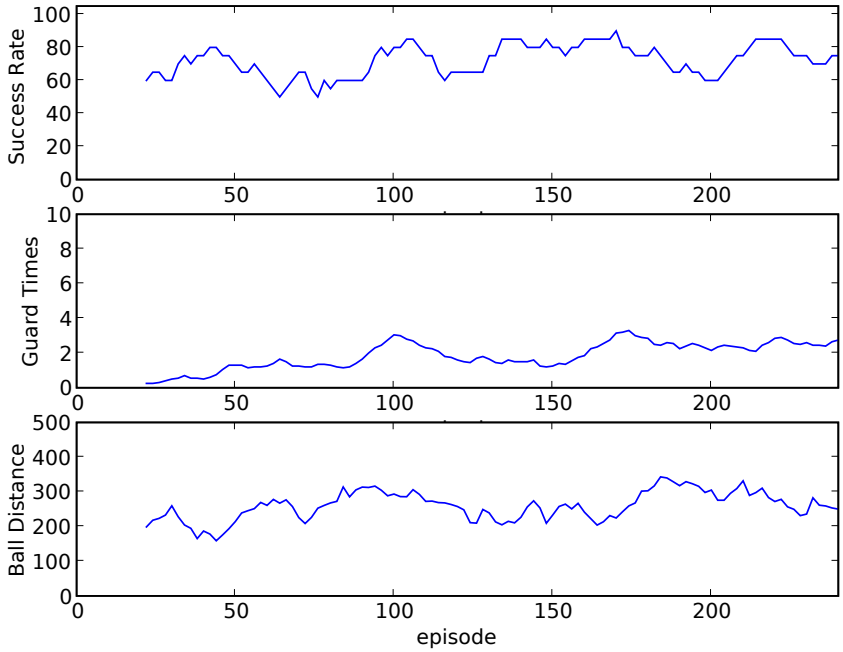


図 5.21: 対称性を利用した拡張現実での学習

トは必要としていないため，一定の分散値のガウシアンノイズをいれるだけにとどめた．

それぞれの入力値に対して次の分散値を用いてノイズを入れた．r には $10mm$, ϕ には $\pi/100rad$,V_v には $3mm$,V_h には $3mm$,X には $10mm$,Y には $10mm$ を分散値としたガウシアンノイズを加えた．学習に関しては対称性を用いた学習方法を用いた，その結果を図 5.22 に示す．ノイズのを入れなかった実験と比較してみると，成功率の上がり方やガード時間， ボールとの距離ともにノイズの入った状況での実験の方が，学習が遅いことがわかる．

さらにノイズを大きくして r には $30mm$, ϕ には $\pi/100rad$,V_v には $10mm$,V_h には $10mm$, X には $30mm$,Y には $30mm$ が分散値としたガウシアンノイズを加えた．ϕ 以外の観測値に先ほどの実験の約 3 倍のノイズを加えた，その結果を図 5.23 に示す．その結果大きなノイズが入っているために学習が十分にできないことが分かる．実験で加えたノイズはボールの距離，自己位置ともに $30mm$，ボールの速度については $10mm$ の分散を持つガウシアンノイズである．実際ロボットで認識した場合，観測値にはさらに大きなノイズが入る．よって実ロボットを使って学習初期からキーパーの学習を行うことは不可能で

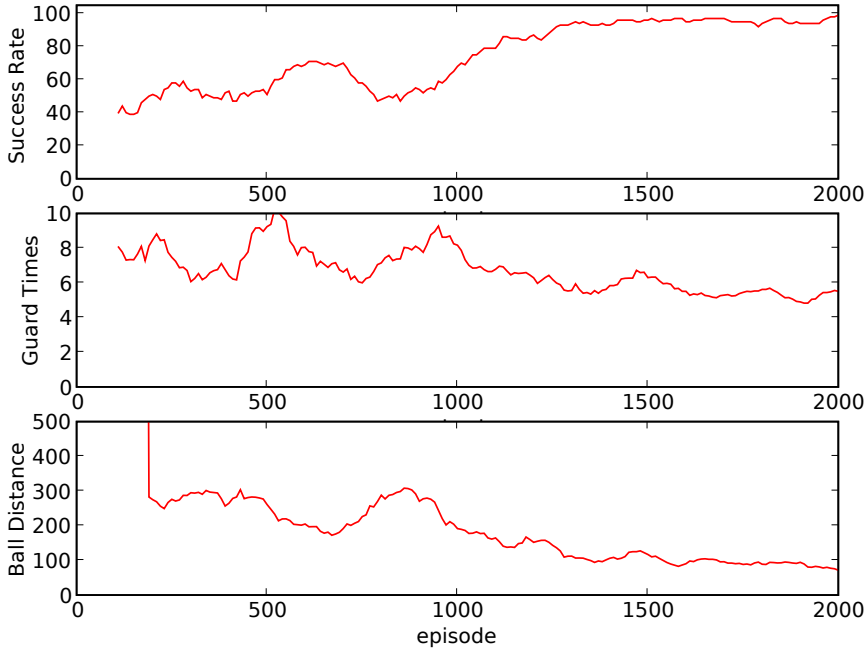


図 5.22: 観測値にノイズ含んだシミュレータでの学習結果

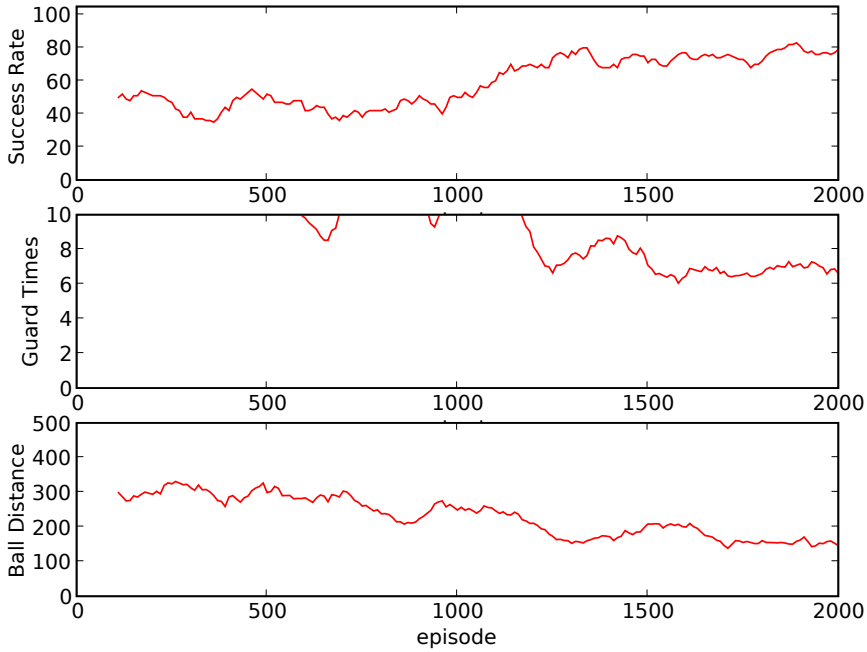


図 5.23: 観測値にノイズを含んだ拡張現実での学習結果．ノイズを大きくした結果学習ができなかった．

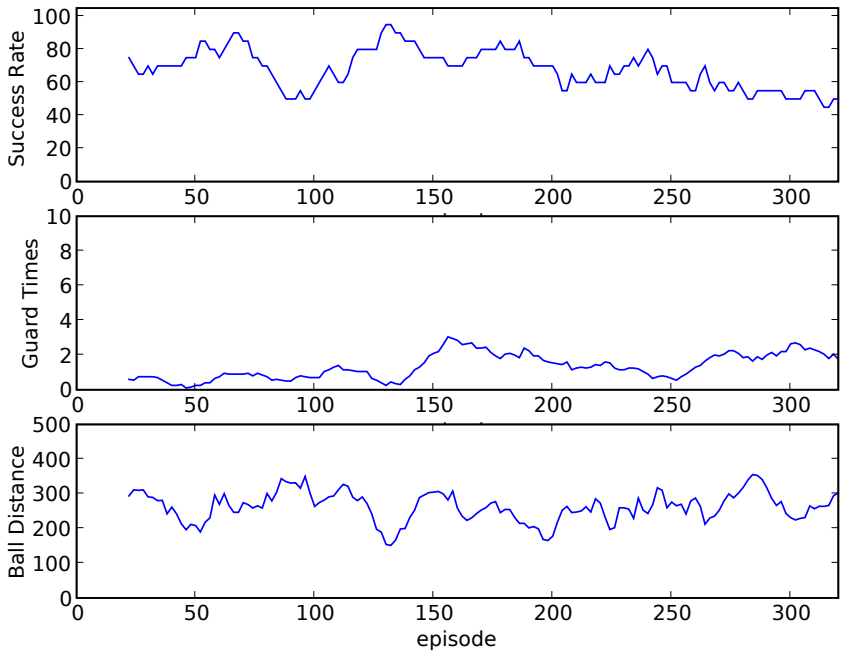


図 5.24: 観測値に微小なノイズを含んだシミュレータでの学習結果を用いての拡張現実での学習結果

あることが分かる．

続いて図 5.22 に示した学習結果を用いて拡張現実での実験を行った，その結果を図 5.24 に示す．図 5.21 と比較しても成功率，ガード回数，ボールの距離ともに大きな変化は無い．そのためシミュレータで観測値に微小なノイズが含まれていても拡張現実での学習には影響無いことがわかる．

5.5 角度を考慮した学習

5.2 では拡張現実の初期状態では位置がずれることをシミュレータの初期状態を原点からずらすことにより，学習済み領域を広げ拡張現実での学習の効率を高めた．拡張現実の初期状態では，位置だけではなく角度も 30 度以下のずれがある．まずはこの事が学習に影響を確かめるために，シミュレータの初期状態で角度をずらした状態から学習を行った，その結果を図 5.25 に示す．この結果より初期状態の角度が影響して，拡張現実と同様にシミュレータでも成功率が 80% までしか向上していないことがわかる．よって拡張現実で行った学習が成功率 80% 以上にならない原因の一つであることを示している．

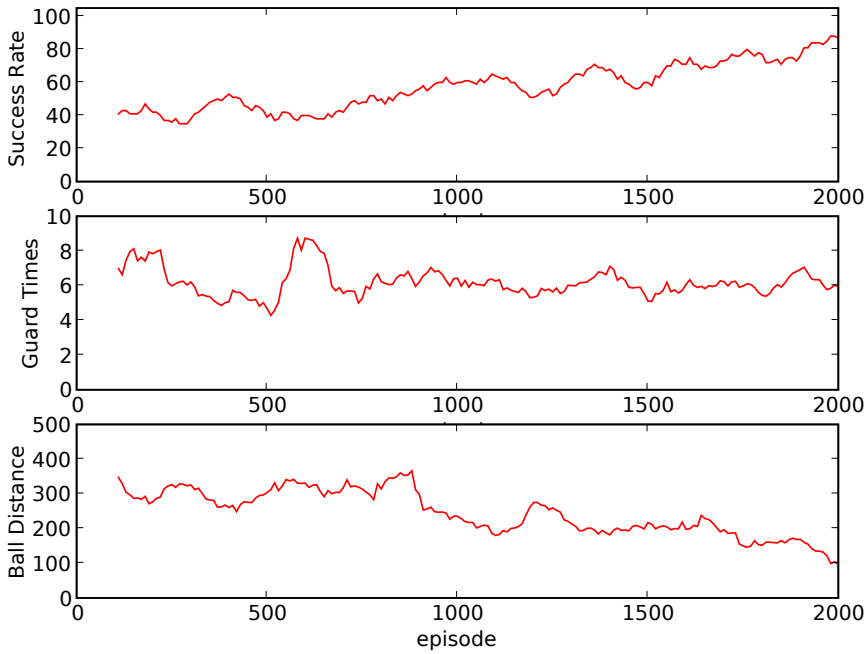


図 5.25: 角度をずらしてのシミュレータでの学習結果

5.6 角度を状態に入力した学習

5.5 に示す通り，初期状態の角度のずれが成功率に関係していることが分かる．初期位置で角度が決められている PK の学習はできたが，ゴールに対して様々な状況でゴールを守らなければならない実践のキーパーに適用することは難しい．そこで学習器に入力する状態にロボットの角度を含める事によって，実践でのキーパーの学習が行えないか実験を行った．ロボットのゴールとの角度を ψ とする．ψ の変域を $[-\pi/4rad, \pi/4rad]$ とした．これを 4.2 で定義した状態の最後に付け足して，7 次元の状態として学習器に入力をした．行動や報酬等はすべて 4.2 で述べたものと同様にした．

その結果を図 5.26 に示す．この結果を見ると状態に角度を入れた時にはシミュレータの学習を行った場合でも，100% の成功率にはいたらず，80% 程度に止まっている．またこの結果を用いて拡張現実で学習を行った結果が図 5.27 である．拡張現実での結果は図 5.21 に比べて，成功率，ガード回数，ボールの距離すべてにおいて，安定して学習している．これは初期位置での角度のずれやロボットの歩行での角度ずれになどを学習器に入力することにより，ロボットの角度に応じた動きができたことがわかる．

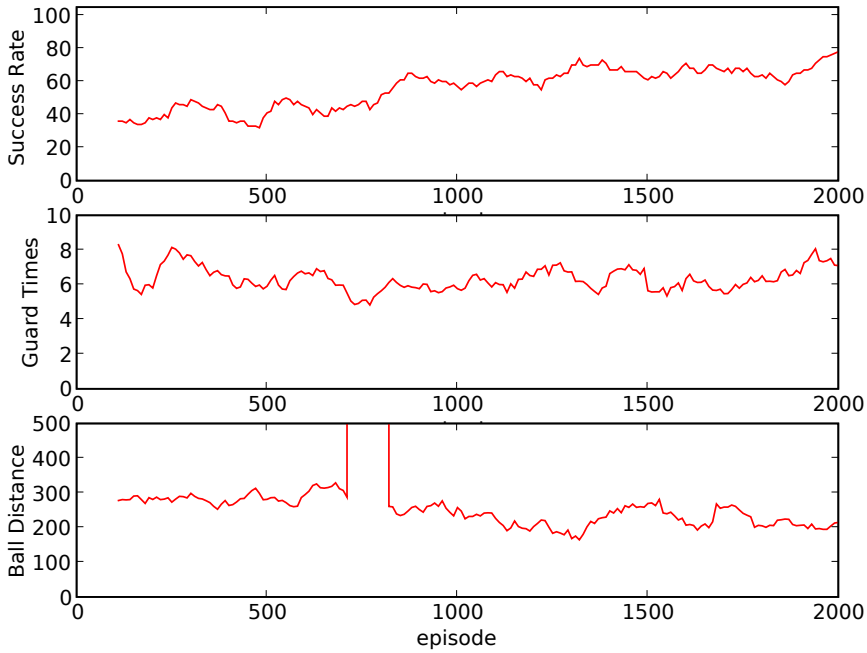


図 5.26: 角度を状態に含めたシミュレータでの学習結果

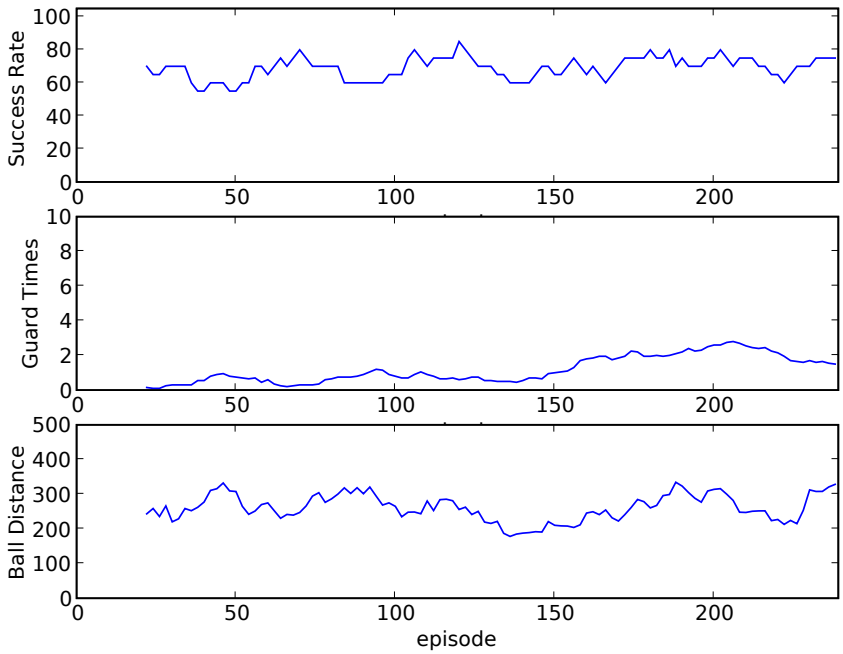


図 5.27: 角度を状態に含めた拡張現実での学習結果

第6章

まとめと今後の課題

本論文ではRoboCupを題材に拡張現実サッカーフィールドシステムの開発を行った．またキーパーの学習を行う際に，シミュレータと実ロボットでの学習の間に拡張現実で学習を行うことによりシミュレータと実ロボットでの環境間の差を少なくし，学習データをシミュレータ，拡張現実と段階的に学習を行うことにより学習の効率を高めた．また現実で行うと人間の労力のかかる学習を，拡張現実で学習を行う事によって，学習が自動的に行われ人間の労力がかからない方法で学習を行うことができた．

本論文ではシミュレータと実ロボットの間の環境として拡張現実「サッカーフィールドシステム」で学習を行った．しかしサッカーフィールドシステムは学習に用いるだけでなく，ロボットへ送信する情報を変えることでロボット開発のデバッグの方法と使用することもできる．さらに実ロボットとサッカーフィールドシステム内のロボットとの対戦をすることによって，シミュレータ環境と現実での動作の違いを確認することができる．

本論文ではPKのキーパーの行動を題材に学習を行ったが，キーパーの学習結果を用いると2体でのキャッチボールの学習の受け手の学習が完了した事になるため，次にパスの出す側のシュートの学習をサッカーフィールドシステムを用いて行う事ができる．またこれらを応用することによって実践でのキーパーの行動の学習や実ロボットでのパスワークの学習の足がかりとなる．

謝辞

はじめに本研究を行うにあたり多大なご指導をいただきました本学大学院情報科学研究科　篠原　歩教授，石野　明助教に深く感謝いたします.

御専門のお立場から的確なご助言を賜りました亀山充隆教授，橋本和夫先生に心から御礼申し上げます.

また学生生活でお世話になりました，篠原研究室の皆さんに深く感謝いたします.

最後に学生生活を暖かく見守って頂いた両親に心から感謝致します.

参考文献

[1] Jolly Pochie. URL: ⟨http://www.jollypochie.org/⟩.

[2] RoboCup Official Site. URL: ⟨http://www.robocup.org/⟩.

[3] RoboCup Simulation League. URL: ⟨http://sserver.sourceforge.net/wiki/index.php/Main_Page⟩.

[4] RoboCup Small Size League. URL: ⟨http://small-size.informatik.uni-bremen.de/⟩.

[5] RoboCup Middle Size League. URL: ⟨http://www.er.ams.eng.osaka-u.ac.jp/robocup-mid/index.cgi⟩.

[6] RoboCup Four-Legged Robot League. URL: ⟨http://www.tzi.de/4legged/⟩.

[7] RoboCup Humanoid League. URL: ⟨http://www.humanoidsoccer.org/⟩.

[8] Hayato Kobayashi, Tsugutoyo Osaki, Eric Williams, Akira Ishino, and Ayumi Shinohara. Autonomous learning of ball trapping in the four-legged robot league. In *RoboCup*, pp. 86–97, 2006.

[9] Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, Vol. 13, No. 3, pp. 165–188, 2005.

[10] 加藤龍憲, 鈴木昭二, 浅田稔. 強化学習によるゴール守備行動の獲得. 第 3 回 JSME ロボメカ・シンポジア講演論文, pp. 37–40, 1998.

[11] Francois Berard. The magic table: Computer-vision based augmentation of a whiteboard for creative meetings. In *IEEE Workshop on Projector-Camera Systems (PROCAM)*, 2003.

[12] 緒方祐介, 有田大作, 菅沼明, 谷口倫一郎. プロジェクタ・カメラシステムを用いたビリヤードの初級者向けショット練習支援. エンターテインメントコンピューティング研究会, pp. 17–23, 2007.

[13] Noriyoshi Shimizu, Naoya Koizumi, Maki Sugimoto, Hideaki Nii, Dairoku Sekiguchi, and Masahiko Inami. A teddy-bear-based robotic user interface. *Comput. Entertain.*, Vol. 4, No. 3, p. 8, 2006.

[14] Nagisa Munekata, Takanori Komatsu, and Hitoshi Matsubara. Marching bear: An interface system encouraging user's emotional attachment and providing an immersive experience. In *ICEC*, pp. 363–373, 2007.

[15] Maki Sugimoto, Georges Kagotani, Minoru Kojima, Hideaki Nii, Akihiro Nakamura, and Masahiko Inami. Augmented coliseum: display-based computing for augmented reality inspiration computing robot. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Emerging technologies*, p. 1, New York, NY, USA, 2005. ACM.

[16] Rodrigo da S Guerra, Joschka Boedecker, Norbert Mayer, Shinzo Yanagimachi, Yasuji Hirosawa, Kazuhiko Yoshikawa, Masaaki Namekawa, and Minoru Asada. Citizen eco-be! league: bringing new flexibility for research and education to robocup. 第25回 SIG-CHALLENGE 研究会, 2007.

[17] PointGrey Research. URL: ⟨http://www.ptgrey.com/⟩.

[18] OpenCV. URL: ⟨http://www.intel.com/technology/computing/opencv/⟩.

[19] GML C++ Camera Calibration Toolbox. URL: ⟨http://research.graphicon.ru/calibration/gml-c-camera-calibration-toolbox-5.html⟩.

[20] M. Piccardi. Background subtraction techniques: a review. *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, Vol. 4, , 2004.

[21] Microsoft Robotics Studio. URL: ⟨http://msdn.microsoft.com/robotics/⟩.

[22] ARAIBO. URL: ⟨http://araibo.mech.chuo-u.ac.jp/⟩.

[23] Haribote Project. URL: ⟨http://araibo.mech.chuo-u.ac.jp/haribote/⟩.

[24] Open Dynamics Engine. URL: ⟨http://www.ode.org/⟩.

[25] OpenGL. URL: ⟨http://www.opengl.org/⟩.

[26] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.